Master in High Performance Computing

Band Parallelization Strategies and Optimization of Real Space Projectors in Quantum ESPRESSO

Supervisor(s):

Pietro Davide Delugas,

Stefano de Gironcoli

Candidate:

Gustavo Paredes Torres

11th edition

2024–2025

Acknowledgments

I would like to express my sincere gratitude to the institutions and people who made this work possible. In particular, I would like to thank the Abdus Salam International Centre for Theoretical Physics (ICTP), the International School for Advanced Studies (SISSA), and the Master's Program in High Performance Computing (MHPC) for their support and for providing an inspiring academic environment.

I am especially grateful to my advisor, Dr. Pietro Davide Delugas, for his guidance, support, and insightful discussions throughout this work.

The research presented in this thesis, as well as my time in Trieste, were supported by ICTP, SISSA, and the Quantum ESPRESSO Foundation (QEF). I am thankful for the resources and opportunities that made this work possible.

I would also like to thank my parents, Marilu Torres and Arturo Paredes, for their unconditional support, and my brothers Ramon, Elizabeth and Arturo for always encouraging me.

I am also grateful to my colleagues and friends, especially Javier Servin, Iván Álvarez, Jhon Moreno, Laura Cabrera and Christian Tica, for their friendship and support during this journey.

Gustavo Paredes Torres
Master in High Performance Computing
Trieste Italy., Wednesday 25th March, 2026

Abstract

QUANTUM ESPRESSO is a widely used open-source package for Density Functional Theory (DFT) calculations. For large systems, its performance is strongly affected by parallel efficiency and load imbalance, particularly in band-parallel and linear-response calculations.

This thesis investigates performance bottlenecks in the `cgsolve_all` solver of the PH.x module, which uses a band-parallel preconditioned conjugate gradient method. In the standard implementation, bands are distributed evenly among band groups using a static scheme. However, the computational cost per band is not uniform, leading to idle time and reduced scalability.

To mitigate this issue, a dynamic band-balancing strategy is proposed. The method measures runtime imbalance and adaptively redistributes bands between groups to reduce waiting time. Performance tests on CPU and GPU architectures demonstrate improved load balancing and stronger scaling behavior, particularly for heterogeneous systems, where performance gains of 20–27% in CPU time were observed under band-parallel scaling. More homogeneous systems demonstrate smaller but consistent improvements.

In addition, this work analyses the role of real-space projectors in the overall computational cost. The section on Optimization of Real Space Projectors identifies key sources of imbalance and communication overhead in projector-related routines. Although full algorithmic restructuring was beyond the available time frame, the problem, main bottleneck and optimization strategies are discussed, including improved data locality, better band–projector mapping, and alternative communication patterns.

Overall, this thesis demonstrates that dynamic workload redistribution and careful analysis of real-space projector routines are effective approaches to improve scalability in band-parallel electronic-structure calculations.

Key words: Quantum Espresso, phonon, bands.

Introduction

Density Functional Theory (DFT) [1] has become one of the most powerful and widely used frameworks in condensed matter physics, quantum chemistry, and materials science.

Over the past decades, several open source and commercial implementations of DFT have been developed to meet the growing demand for large-scale simulations in both academic and industrial contexts. Among them, QUANTUM ESPRESSO (QE) [2,3] is one of the most versatile and community-driven software packages in this field, offering a comprehensive suite of codes for electronic structure calculations and materials modeling based on plane wave pseudopotential methods. QUANTUM ESPRESSO stands for Quantum opEn Source Package for Research in Electronic Structure, Simulation, and Optimization. It is a distribution (an integrated suite) of software for atomistic calculations based on electronic-structure theory, using density functional theory, a plane-wave basis set, and pseudopotentials. To improve the efficiency of these costly processes, the QE suite offers parallel computing techniques targeting multicore and distributed systems. This hybrid parallelism, based on the effective use of OpenMP, the Message Passing Interface (MPI), the Fast Fourier Transform library (FFTW) [4], and the Linear Algebra Subprograms (LAPACK) [5], ScaLAPACK [6], or ELPA [7], support to run in GPU using OpenACC, making QE a high-performance computing library.

It is freely available under the terms of the GNU General Public License (GPL) [8], enabling accurate predictions of structural, electronic, and vibrational properties of materials at a computationally feasible cost. QE codes are constructed around the use of periodic boundary conditions and efficient convergence criteria to make the treatment of infinite crystalline systems straightforward. QE can be used for any crystal structure or supercell, including metals and insulators. The framework includes all standard exchange-correlation functionals such as LDA and GGA [9], as well as advanced functionals and generalizations of the Kohn-Sham approach such as meta-GGA [10], Hybrid functional, and Hubbard U corrections. Using these various levels of theory, Quantum ESPRESSO can perform a wide range of calculations on material properties. Examples include:

- Vibrational properties with ph.x, including phonon spectra, phonon dispersions, phonon density of states, force constants, vibrational frequencies, and the assessment of dynamical crystal stability.
- Electronic and structural properties with pw.x, such as total energy, equilibrium geometry, lattice constants, pressure, and the stress tensor.

In this thesis, we will focus on two of the main modules of the suite: `PH` which performs calculations for phonons and related vibrational and dielectric properties using density functional perturbation theory [2] and `PWscf` code which performs the Kohn-Sham self-consistent calculations, offering KS orbitals and energies for isolated or extended/periodic systems and then with that one can do the computation of total energy and forces. `PH` module is particularly important because it establishes the theoretical and computational foundation for many other QE core modules, and, by building on this modularity, improvements introduced therein can be extended to the other modules. The central focus of this thesis will be the detection, real-time diagnosis, and mitigation of workload imbalances arising during the execution of these two applications. We will leverage the iterative structure of their algorithms, which allows workload monitoring and redistribution to be interleaved with computation, enabling the system to converge toward a steady, minimized imbalance distribution throughout the procedure. Adopting this approach, the first part of the work has focused on improving the parallel efficiency of the band parallelism in the linear response solver (CGSOLVE_ALL.F90), by developing and implementing a dynamic band-balancing strategy to mitigate load imbalance among band groups in large-scale simulations.

Code availability. The implementation developed in this work, including the dynamic band-balancing modifications to `cgsolve_all` and the associated performance analysis tools, is maintained in a GitLab repository. The repository contains the modified source files for Quantum ESPRESSO, benchmarking scripts, and the post-processing tools used to generate the figures and performance metrics presented in this thesis.

Making the code publicly available ensures reproducibility of the results and provides a foundation for further research and optimization of band-parallel algorithms in Quantum ESPRESSO.

https://gitlab.com/gparedes137/q-e/-/tree/dynamic-band-balancing

Contents

List of Figures

List of Tables

Chapter 1

From the many-body problem to the functional density theory

1.1 Introduction

Quantum ESPRESSO [3] is an integrated suite of Open Source computer codes for electronic structure calculations and materials modelling at the nanoscale. It is based on density functional theory, plane-wave, and pseudopotential methods [11, 12]. We will first very briefly sketch the main points of the theoretical background, introducing DFT, plane-waves and pseudopotentials. And then, to understand the subroutines we will be discussing, we first will describe with more detail some of the theoretical and computational aspects relative to pseudopotentials and linear Density Functional Perturbation Theory (DFPT); for this latter we will mostly refer to the review of DFPT in [2].

1.2 Theory approach

Although ordinary non-relativistic quantum mechanics is, in principle, sufficient to account for the vast majority of phenomena in solid-state physics, its direct application to real materials is computationally intractable. As P. A. M. Dirac famously observed in 1929, "The fundamental laws necessary for the mathematical treatment of a large part of physics and the whole of chemistry are thus completely known. . . " but he immediately added that ". . . the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble". This trade-off between theoretical completeness and practical impossibility is precisely what motivates the hierarchy of approximations used in modern electronic-structure theory. In order to make materials-property calculations feasible, one systematically introduces well-founded simplifications—each reducing the complexity of the many-body problem while retaining the essential physics. The following sections provide a brief sketch of these approximations and the rationale that underlies them.

Solid-state physics begins with the solution of the Schrödinger equation for a system of many electrons and nuclei. The Hamiltonian contains kinetic, Coulombian interactions for the electron and nuclei, electron-electron interactions, nuclear-nuclear interactions, electron-nuclear interactions, and external potential terms. These interactions can be expressed in the following Hamiltonian:

$$H = -\sum_i \frac{1}{2}\nabla_i^2 - \sum_I \frac{1}{2M_I}\nabla_I^2 + \sum_{\substack{i,j \\ i\neq j}} \frac{1}{2}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} - \sum_{i,I} \frac{Z_I}{|\mathbf{R}_I - \mathbf{r}_i|} + \sum_{\substack{I,J \\ I\neq J}} \frac{1}{2}\frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|}, \tag{1.1}$$

this equation is in atomic units ($\hbar = |e| = m_e = \frac{1}{4\pi\epsilon_0} = 1$, $E = 27.21eVs$) where $\mathbf{R_I}$ is the coordinate of the Ith nucleus, M_I its mass, and r_i is the coordinate of the ith electron, Z refers to the charge of the nuclei (I) or electron (i). This is known as many body Hamiltonian *many-body problem*. Various approximations were adopted to solve this many-body Hamiltonian.

Its exact solution is unattainable due to the enormous number of degrees of freedom. To make it manageable, a number of approximations are introduced:
Born-Oppenheimer (BO): Since nuclei are much heavier than electrons, their velocities are much smaller in comparison, the Schrödinger equation can be separated into two parts: one part describes the electronic wavefunction for a fixed nuclear geometry, the second describes the nuclear wavefunction. Here, it is assumed that the electrons move in an electrostatic field generated by the nucleus. Since the nucleus is considered to be at rest, the Hamiltonian (1.1) can be rewritten as follows:

$$H = -\sum_i \frac{1}{2}\nabla_i^2 + \sum_{\substack{i,j \\ i\neq j}} \frac{1}{2}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} - \sum_{i,I} \frac{Z_I}{|\mathbf{R}_I - \mathbf{r}_i|} + \sum_{\substack{I,J \\ I\neq J}} \frac{1}{2}\frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|}, \tag{1.2}$$

In the BO approximation, the total wavefunction is limited to a single electronic surface, i.e. a specific electronic state. While the BO approximation is usually very accurate, it breaks down when two or more electronic states are close in energy at specific nuclear geometries.

Hartree and Hartree-Fock (HF): describe electrons moving in a medium potential, incorporating (in HF) the Pauli exclusion principle by means of Slater determinants, but without fully capturing the electronic correlations. For non-interacting particles, the wave function can be written as a product of individual-particle wave functions. Hartree considered each system obeying a Schrödinger equation, and one can write the wave equation for an n particle system as:

$$\psi(\mathbf{r}_i) = C_N \prod_i^n \phi(\mathbf{r}_i), \tag{1.3}$$

The interaction of one electron with the others is incorporated on average. The Schródinger equation can be written as:

$$\left(\frac{1}{2}\nabla^2 + V_{ext}(\mathbf{r}) + V_{additional}(\mathbf{r})\right)\phi_i = \epsilon_i \phi_i, \tag{1.4}$$

where $V_{ext}(\mathbf{r})$ is the average potential felt by the electron at $\mathbf{r}$ and $V_{additional}(\mathbf{r})$ is related to electron electron interaction. This is the Hartree approximation, in this approximation, correlations are not considered and the wave function is not antisymmetric, etc.

To better describe a wave function, antisymmetry must also be included. The Slater determinant is useful for this purpose as it accounts for spin. Interchanging the positions of two electrons is equivalent to interchanging the corresponding columns. If two electrons with the same spin interchange positions, ψ^D, the determinant must change sign. This is known as the exchange property and involves the manipulation of the Pauli principle. One of the most common approaches to the problem of many fermions is to consider each electron separately in the one-electron approximation.

$$\psi^D(\mathbf{r}) = \frac{1}{\sqrt{N!}}\begin{vmatrix} \chi_1(\mathbf{r}_1) & \chi_2(\mathbf{r}_1) & \dots & \chi_n(\mathbf{r}_1) \\ \chi_1(\mathbf{r}_2) & \chi_2(\mathbf{r}_2) & \dots & \chi_n(\mathbf{r}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1(\mathbf{r}_n) & \chi_2(\mathbf{r}_n) & \dots & \chi_n(\mathbf{r}_n) \end{vmatrix}, \tag{1.5}$$

$$H = \left(-\sum_i \frac{1}{2}\nabla_i^2 + V_{ext}(\mathbf{r}_i) \right) + \frac{1}{2} \sum_{\substack{i,j \\ i \neq j}} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}, \tag{1.6}$$

with this determinant as wave function solving the Schrödinger we arrive to:

$$\left(-\frac{1}{2}\nabla^2 + V_{ext} + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} dr' - \frac{1}{2} \sum_{i,j,\sigma} \int \frac{\phi^*_{j,\sigma}(\mathbf{r}')\phi_{i,\sigma}(\mathbf{r}')\phi_{j,\sigma}(\mathbf{r})}{\phi_{i,\sigma}(\mathbf{r})|\mathbf{r} - \mathbf{r}'|} dr\ dr' \right) \phi_{i,\sigma}(\mathbf{r}) = \epsilon_i \phi_{i,\sigma}(\mathbf{r}), \tag{1.7}$$

This is the Hartree–Fock equation, which describes non-interacting electrons under the influence of a mean-field potential consisting of a classical Coulomb potential and a nonlocal exchange potential. The correlation energy accounts for the reduction in energy due to quantum fluctuations. As this is still a $3N$-dimensional problem, an analytical solution is challenging to obtain. Although this method succeeded in describing various systems, it failed because of the poor treatment of electron exchange and correlation.

Hartree-Fock provides a feasible approximation of both the ground density and the energy and constitutes the starting point for very accurate quantum-chemistry approaches such as configuration interaction and other advanced methods (see references [13, 14]).

Another major computational and conceptual simplification came with the Density Functional Theory (DFT), formulated by Hohenberg and Kohn (1964) [15]. This theory reduces the many-body problem to a formulation expressed in terms of the ground-state density as the fundamental quantity, rather than the many-electron wave function. This reduces the problem's complexity from $3N$ variables to 3. The electron density, $\rho(\mathbf{r})$, can be expressed as follows:

$$\rho(\mathbf{r}) = N \int d^3\mathbf{r}_2 \cdots d^3\mathbf{r}_N \psi(\mathbf{r}, \mathbf{r}_2, \cdots, \mathbf{r}_N)\psi^*(\mathbf{r}, \mathbf{r}_2, \cdots, \mathbf{r}_N) \tag{1.8}$$

The idea that density could be used as a fundamental parameter, just as wave functions are, was first suggested by L. Thomas and E. Fermi in 1927.

The correspondence between the wave functions and the ground state density is established by the two Hohenberg–Kohn theorems, which state that:

- For any system of interacting particles in an external potential $V_{ext}(\mathbf{r})$, the ground state density $n_{GS}(\mathbf{r})$ uniquely determines the potential. This means that the electron density uniquely determines the Hamiltonian operator. In other words there is a one to one correspondence between the electron density $n_{GS}(\mathbf{r})$ and external potential V_{ext}.

- As a variational principle: for a given V_{ext} the functional energy $E[n]$ reaches its minimum in the fundamental density of the system, which defines the fundamental energy. In mathematical terms:

$$E_{HK}[n] = F_{HK}[n] + \int d^3\mathbf{r} V_{ext}(\mathbf{r}) n(\mathbf{r}), \tag{1.9}$$

$$\left. \frac{\delta E_{HK}[n]}{\delta n(\mathbf{r})} \right|_{n=n_{GS}} = 0, \tag{1.10}$$

  $F_{HK}[n]$ is the universal Hohenberg-Kohn functional, which includes contributions from both the kinetic energy and the many-body interaction energy of the electrons. This makes it essentially impossible to compute $F_{HK}[n]$ exactly.

1.3 The Kohn-Sham approach

The universal functional $F[n]$ containing kinetic and interaction energy is unknown, so Kohn and Sham (1965) introduced a fictitious system of non-interacting electrons subject to an external effective potential V_{KS} that depends on the electron density and reproduces the same density as the real system, defining the functional for a non-interacting system:

$$E_{KS}[n] = T_s f[n] + E_H[n] + E_{XC}[n] + \int d^3\mathbf{r} V_{ext}(\mathbf{r}) n(\mathbf{r}), \tag{1.11}$$

where: $T_s[n]$ is the kinetic energy of the non-interacting system, $E_H[n]$ is the classical interaction of Coulomb (Hartree energy) with the potential associated:

$$V_H(\mathbf{r}) = \frac{\delta E_H[n]}{\delta n(\mathbf{r})} = \frac{e^2}{2} \int d^3\mathbf{r}' \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \tag{1.12}$$

and $E_{XC}[n]$ is the exchange and correlation function, which is simply the sum of the error made in using a non-interacting kinetic energy and the error made in treating the electron-electron interaction classically.

The ground state density of the fictitious system that minimises the energy (1.11) is assumed to be the same as that of the interacting system described by (1.9). The problem does not appear to be simplified by this procedure, since the many-body interaction remains. However, compared with the Hohenberg-Kohn method, this approach separates the contributions of the kinetic energy and long-range Coulomb energy from the F_{HK} functional. This separation enables E_{xc} to be expressed as a local functional based on reasonable physical assumptions. An introductory review of the various functionals present can be found in the literature [11]. The expression of the energy (1.1) corresponds to the energy of a gas of non-interacting electrons moving in an external potential $V_{KS}(\mathbf{r})$:

$$E_{KS}[n] = T_s[n] + \int d^3\mathbf{r} V_{KS}(\mathbf{r}, [n]) n(\mathbf{r}), \tag{1.13}$$

where the density-dependent Kohn-Sham potential is defined as:

$$V_{KS}(\mathbf{r}, [n]) = V_H(\mathbf{r}) + V_{ext}(\mathbf{r}) + V_{XC}(\mathbf{r}), \tag{1.14}$$

where $V_XC(\mathbf{r})$ is the exchange correlation potential:

$$V_{xc}(\mathbf{r}) = \frac{\delta E_{XC}[n]}{\delta n(\mathbf{r})}. \tag{1.15}$$

Since the fictitious system consists of non-interacting electrons, the density and the kinetic energy can be expressed in terms of single particle wave functions ψ_i, known as Kohn–Sham orbitals:

$$T_s = -\frac{\hbar^2}{2m} \int d^3\mathbf{r} \psi_i^*)(\mathbf{r}) \nabla^2 \psi_i(\mathbf{r}), \qquad n(\mathbf{r}) = \sum_i ||\psi_i(\mathbf{r})||^2. \tag{1.16}$$

Minimizing the energy (1.13) With respect to the wave function, ψ_i^*, it leads to the Kohn–Sham equation for ψ_i:

$$H_{HS}\psi_i(\mathbf{r}) = \left(-\frac{\hbar^2}{2m}\nabla^2 + V_{KH}(\mathbf{r}, [n])\right)\psi_i(\mathbf{r}) = \epsilon_i \psi_i(\mathbf{r}) \tag{1.17}$$

Since the Kohn–Sham potential (1.14) depends on the electron density, which depends on the wave functions via (1.16), equation (1.17) must generally be solved iteratively. This set of nonlinear equations describes the behaviour

of non-interacting "electrons" within an effective local potential. For an exact functional and, consequently, an exact local potential, the "orbitals" yield the exact ground state density. The Kohn-Sham equations have the same structure as the Hartree-Fock equations, except that the nonlocal exchange potential is replaced by the local exchange correlation potential, V_{XC}.

1.4 Numerical approach for the Kohn-Sham equation

The effective potential of a system's ground state can be constructed if an initial guess of the density $\rho(\mathbf{r})$ is provided. This potential can be expressed as follows:

$$V_{eff} = V_H + V_{ext} + V_{XC}, \tag{1.18}$$

Once V_{eff} is defined, the Kohn–Sham Hamiltonian can be constructed and the Kohn–Sham equations can be solved for a given set of trial wave functions. The resulting wave functions are used to compute a new electronic density. The procedure is repeated iteratively until the input and output densities (or potentials) are consistent within a specified tolerance. At this point the system is said to have reached *self-consistency*.

The initial density is typically constructed from atomic charge densities and depends on the atomic positions and the crystal structure. In the self-consistent field (SCF) procedure, solving the Kohn–Sham Hamiltonian amounts to computing its eigenvalues and eigenvectors. The numerical approach used to obtain these eigenpairs depends strongly on the basis set employed.

For methods based on localized basis sets, such as Gaussian orbitals or numerical atomic orbitals, the Hamiltonian matrix is relatively small and can often be constructed explicitly and diagonalized using standard dense linear-algebra routines. In contrast, plane-wave methods employ a very large basis set, which makes the explicit construction and full diagonalization of the Hamiltonian matrix impractical. Instead, iterative diagonalization methods are used to obtain only the lowest eigenstates required for the electronic structure calculation.

Quantum ESPRESSO adopts the plane-wave basis representation, and therefore relies on iterative solvers such as conjugate-gradient or Davidson-type algorithms to compute the Kohn–Sham eigenstates efficiently. These iterative methods avoid the explicit construction and diagonalization of the full Hamiltonian matrix while exploiting the structure of the plane-wave representation.

The eigenvalues and eigenvectors obtained from this numerical procedure can be used to compute relevant physical quantities such as the total energy, forces acting on the atoms, and the stress tensor. The overall workflow of the DFT self-consistent procedure is illustrated in Fig. 1.1. The DFT algorithm is as follows:

- Start with a initial guess for the density $\rho(\mathbf{r})$.
- Compute $V_{KS}(\mathbf{r})$.
- Solve KS equation (1.17) by diagonalizing the Hamiltonian H_{KS}, obtaining $\{\psi_i(\mathbf{r})\}_i, \epsilon_i$.
- Compute the electron density using Eq. (1.16).
- Repeat until the density converges.

Representing the operator $\hat{H}_{KS}$, the KS orbitals, the density and then solving the KS equations at each given density remains computationally challenging. A practical strategy is needed to solve it efficiently. Many different strategies have been implemented [11], all of which start from the choice of different basis sets for representing the orbitals and consequently adapting the algorithms.

Quantum ESPRESSO uses the Plane-Wave plus pseudo potential (PW+PP) method, which consists of represent the orbital as superposition of plane waves $\frac{1}{\sqrt{V_0}} e^{i\mathbf{G}\cdot\mathbf{r}}$, where $\mathbf{G}$ is any vector belonging to the reciprocal lattice of the periodic system, and thus projecting Eq.(1.17) on the basis of plane waves defined on the lattice [12]. Plane waves constitute a widely used basis set for electronic structure calculations, as they naturally adapt to the periodic boundary conditions of crystalline systems. The main advantage of plane-wave basis sets is that this set is increasingly accurate

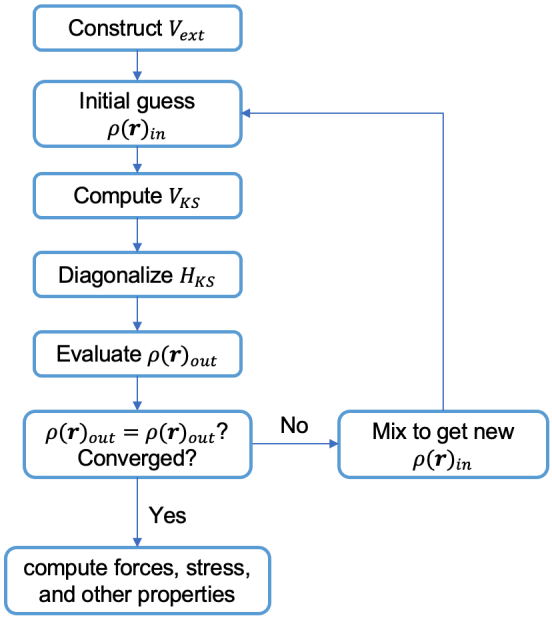


Figure 1.1: `PW` self-consistency loop for solving Kohn-Sham equations.

depending on how many plane-waves are included in the basis set and such increasing accuracy is described by one single parameter that is conventionally indicated by the kinetic-energy cutoff of plane waves, which in Rydberg units is simply the squared modulus of the the wave vector **G** of the plane wave. The main drawback of such a basis would be the fact that one would need a very large number of plane waves to describe with the necessary accuracy the atomic orbitals in the core regions around each atom. For this reason, the usage of plane wave is almost always coupled with the adoption of the pseudopotential approximation.

1.4.1 Pseudopotentials in Density Functional Theory

Pseudopotentials provide an essential strategy for removing tightly bound core electrons from explicit consideration while preserving the relevant scattering properties of valence states. In the following, we adopt the framework for pseudopotentials as presented in Chapter 11 of Martin's *Electronic Structure: Basic Theory and Practical Methods* [11].

As emphasised by Martin, this reduction not only lowers the computational cost but also preserves the key physical scattering properties required for accurate total-energy and band-structure calculations.

A central class of pseudopotentials is the family of *norm-conserving pseudopotentials* (NCPPs). Here, each pseudo-wavefunction is constructed to match its all-electron counterpart outside a chosen core radius while conserving the radial norm for each angular-momentum channel. These constraints ensure reliable transferability across diverse bonding environments and accurate reproduction of atomic reference data.

A key methodological advance presented is the *Kleinman–Bylander (KB) separable representation* of the non-local part of the pseudopotential. By expressing the angular-momentum–dependent non-local operator as a sum of separable projectors,

$$\hat{V}_{\mathrm{NL}} = \sum_{l} |\chi_l\rangle \; D_l \; \langle\chi_l| \; ,$$

with projectors $|\chi_l\rangle$ derived from atomic pseudo-valence states, the KB formulation greatly simplifies the application of non-local operators in plane-wave calculations. Martin highlights that this separable form reduces the computational effort to a small number of projector overlaps, enabling efficient large-scale DFT and molecular dynamics simulations. Care must nonetheless be taken to avoid "ghost states" that may arise from poorly chosen

projectors.

Ultrasoft Pseudopotentials (USPPs), relax the strict norm-conservation constraint of NCPPs. By allowing softer pseudo-wave functions inside the core region, USPPs further reduce the plane-wave cutoff required for convergence, improving efficiency for transition metals and other systems with localised valence states.

USPPs extend the norm–conserving framework by relaxing the norm-conservation constraint inside atom-centred augmentation spheres. To compensate for the loss of norm, the USPP formalism introduces a set of *augmentation functions* $Q_{ij}(\mathbf{r})$ together with a non-trivial overlap operator,

$$S = \mathbb{1} + \sum_{ij} |\beta_i\rangle Q_{ij} \langle\beta_j|,$$

where the projector functions β_i span the augmentation region. These augmentation terms reconstruct the correct charge distribution in the core region while allowing the pseudo–wave functions to be significantly smoother than their norm-conserving counterparts.

The Projector Augmented-wave (PAW) method represents a further unifying framework that combines the computational efficiency of pseudopotentials with the formal completeness of all-electron methods. Central to PAW is the idea that the true all-electron Kohn–Sham wavefunction $|\Psi\rangle$ can be reconstructed exactly from a smooth pseudo-wavefunction $|\tilde{\Psi}\rangle$ through a linear transformation,

$$|\Psi\rangle = \hat{T}|\tilde{\Psi}\rangle = |\tilde{\Psi}\rangle + \sum_i \left(|\phi_i\rangle - |\tilde{\phi}_i\rangle\right)\langle\beta_i|\tilde{\Psi}\rangle,$$

where the partial waves $|\phi_i\rangle$ and $|\tilde{\phi}_i\rangle$ coincide outside atom-centred augmentation spheres but differ inside them, and where the localised projectors β_i span the augmentation region. This augmentation construction ensures that the all-electron nodal structure and the correct charge density in the core region is fully restored, while retaining the smoothness of the pseudo-wavefunction in the interstitial region. As in the case of USPP, the PAW formalism introduces corresponding augmentation densities and multipole corrections inside each sphere, constructed so that all augmentation contributions vanish outside the spheres. This guarantees accurate total energies, forces, and stresses, while keeping the plane-wave cutoffs comparable to those required for ultrasoft pseudopotentials. As such, PAW represents the most accurate member of the augmentation-based pseudopotential family, achieving all-electron fidelity with computational cost close to that of USPPs.

Although the KB formulation stems primarily from norm-conserving potentials, USPPs and PAWs may be viewed, from an implementation standpoint, as part of the same broader family of separable pseudopotentials that extends NCPPs.

The Kleinman–Bylander separable form, and the ultrasoft extension constitutes the core pseudopotential method used in Quantum ESPRESSO and represents the general form of the non-local part of the Kohn-Sham Hamiltonian, to which we will refer in the remainder of this thesis.

1.4.2 Sternheimer Self-Consistent Linear Response

Within QE, the linear responses of the Kohn–Sham system to a external perturbations are computed by first-order perturbation theory within the Sternheimer framework [2]. Given an external perturbing potential ΔV, the objective is to determine the first-order correction to the Kohn–Sham orbitals, $\Delta\psi_i$, and the induced self-consistent variation of the potential, ΔV_{SCF}.

Linear system for the perturbed orbitals. The perturbed Kohn–Sham orbitals satisfy a linear equation of the form

$$(H_{\mathrm{KS}} - \varepsilon_i)\,\Delta\psi_i = -Q^{\mathrm{c}}\,\Delta V_{\mathrm{SCF}}\,\psi_i, \tag{1.19}$$

where H_{KS} is the unperturbed Kohn–Sham Hamiltonian, ε_i the reference eigenvalue, and Q^{c} projects onto the unoccupied (conduction) subspace. Equation (1.19) is the Sternheimer equation obtained by linearising the Kohn–Sham equations around the ground state [2].

Density response from first-order orbitals. Once (1.19) is solved for all occupied states, the induced charge density follows directly from the first-order orbitals:

$$\Delta n(\mathbf{r}) \;=\; 2\sum_{i} \Re\left[\psi_i^{*}(\mathbf{r})\,\Delta\psi_i(\mathbf{r})\right], \tag{1.20}$$

which is the linearized form used in DFPT [2].

Update of the self-consistent perturbing potential. The self-consistent first-order Kohn–Sham potential is rebuilt from the induced density, starting from the bare perturbation:

$$\Delta V_{\mathrm{SCF}} \;=\; \Delta V \;+\; \Delta V_{\mathrm{H}}[\Delta n] \;+\; \Delta V_{\mathrm{xc}}[\Delta n], \tag{1.21}$$

where the second and third terms are the linear Hartree and exchange–correlation responses generated by Δn [2].

Self-consistent Sternheimer loop (two nested iterations). Equations (1.19)–(1.21) define a self-consistent fixed-point problem for $(\Delta\psi_i, \Delta n, \Delta V_{\mathrm{SCF}})$. We iterate:

1. **Initialize:** $\Delta V_{\mathrm{SCF}}^{(0)} \leftarrow \Delta V$.
2. **Linear solve:** For the current $\Delta V_{\mathrm{SCF}}^{(p)}$, solve (1.19) for $\Delta\psi_i^{(p)}$ using an iterative solver with the conduction-manifold projector Q^{c}.
3. **Density update:** Build $\Delta n^{(p)}$ via (1.20).
4. **Potential update:** Construct $\Delta V_{\mathrm{SCF}}^{(p+1)} = \Delta V + \Delta V_{\mathrm{H}}[\Delta n^{(p)}] + \Delta V_{\mathrm{xc}}[\Delta n^{(p)}]$ and mix until convergence.

At convergence, ΔV_{SCF}, $\Delta\psi_i$, and Δn are mutually consistent, and linear-response quantities (forces, dynamical matrices, dielectric responses, electron–phonon couplings, *etc.*) can be evaluated [2].

In the following chapter we focus on the performance of the conjugate-gradient solver implemented in the routine `cgsolve_all`, which is responsible for computing the Kohn–Sham eigenstates within the iterative diagonalization procedure used by `phonon.x`.

Chapter 2

Band Parallelism in PHonon

2.1 Algorithms relevant for this work

The iterative diagonalization can be done using a block Davidson method or the Conjugate Gradient (CG) method. This diagonalization is performed for each k-point. The number of occupied KS orbitals is determined by the number of electrons in the unit cell. Most of the computational time is spent on iterative diagonalization and density calculation, with the remainder on initialization and post-processing routines.

2.1.1 Dual-Space Application of the Kohn–Sham Hamiltonian

The diagonalisation of the Kohn–Sham Hamiltonian is performed using a dual-space technique, in which real-space and reciprocal-space representations are used synergistically to optimize different components of the Hamiltonian. The Kohn–Sham potential is split into a *local* part and a *non-local* part. The local contribution,

$$V_{\mathrm{loc}}(\mathbf{r}) \;=\; V_{\mathrm{H}}(\mathbf{r}) \;+\; V_{\mathrm{xc}}(\mathbf{r}) \;+\; V_{\mathrm{loc}}^{\mathrm{ps}}(\mathbf{r}),$$

comprising the Hartree, exchange–correlation, and local pseudopotential terms, is most efficiently applied in real space as a simple pointwise multiplication between the potential and the wave function,

$$\phi_i(\mathbf{r}) \;\mapsto\; V_{\mathrm{loc}}(\mathbf{r})\,\phi_i(\mathbf{r}).$$

The non-local component of the Hamiltonian is written in the separable Kleinman–Bylander form,

$$\hat{V}_{\mathrm{NL}} = \sum_{lm} |\beta_l\rangle\, D_{lm}\, \langle\beta_m|,$$

so that the action of $\hat{V}_{\mathrm{NL}}$ on a Kohn–Sham orbital $|\psi_i\rangle$ can be decomposed into two steps:

$$\langle\beta_m|\psi_i\rangle \qquad \text{followed by} \qquad \hat{V}_{\mathrm{NL}}|\psi_i\rangle = \sum_{lm} |\beta_l\rangle\, D_{lm}\, \langle\beta_m|\psi_i\rangle.$$

The evaluation of the projector–wave function overlaps $\langle\beta_m|\psi_i\rangle$ (the `calbec` step in `pw.x` code) is traditionally performed in reciprocal space, where the plane-wave cutoff provides a natural truncation of the basis and hence reduces the total number of operations.

For very large simulation cells, however, reciprocal-space evaluation becomes progressively more expensive because the number of plane waves scales with the volume of the simulation domain. In such cases, it is often more efficient to evaluate the projector overlaps directly in real space. This takes advantage of the strict real-space localization of the projectors $|\beta_m(\mathbf{r})\rangle$, whose support is restricted to compact regions around each atom, thereby reducing the computational workload and improving scaling for large systems.

Chapter {3} it is dedicated to the analysis of these reciprocal and real space projections.

2.1.2 Preconditioned Conjugate Gradient (PCG) Method in `cgsolve_all`

One of the most computationally intensive components of the PWscf code during self-consistent electronic structure calculations is the solution of the generalised eigenvalue problem:

$$H\psi_i = \epsilon_i S\psi_i, \qquad i = 1, \ldots, N, \tag{2.1}$$

where H is the Kohn-Sham Hamiltonian, S is the overlap matrix (Identity in plane wave pseudopotential calculations), and N is the number of occupied Kohn-Sham states. Eigenvectors satisfy the generalized orthonormality condition $\langle\psi_i|S|\psi_j\rangle = \delta_{ij}$.

Quantum ESPRESSO offers two main iterative solvers for this task: (i) the Davidson algorithm, which is fast and robust but memory demanding (due to the subspace expansion), and (ii) the Conjugate Gradient (CG) method, which is more memory efficient since it updates one band at a time by minimising the Rayleigh quotient [1]. In this context a band represents an allowed electronic state of the crystal system. In an isolated atom, electrons occupy discrete levels (1s, 2s, 2p, etc.), when many atoms come together to form a solid, those levels are mixed, split up and form very dense sets of levels. These continuous sets of levels are called electronic bands. In other words, it is a system-wide quantum state obtained by solving the Kohn-Sham equations, where each solution is a wave function with an associated energy. In a crystal, each band exists at each point k of the reciprocal space, so we are actually talking about band n at point k.

Although this chapter of the thesis concentrates on the PHonon library [16], particularly the `ph.x` executable [17], it is important to note that phonon perturbations require the solution of linear response equations of the form:

$$(H - \epsilon_i + Q)\, d\psi_i = d0\psi_i, \tag{2.2}$$

where Q is a projection operator removing components along the occupied manifold. These linear equations are not eigenvalue problems, but *shifted linear systems* that must be solved independently for each band i. The subroutine responsible for solving Eq. (2.2) is `cgsolve_all.f90`, located in folder `q-e/LR_Modules/`, which implements a *classical preconditioned conjugate gradient (PCG)* algorithm tailored for linear response density functional perturbation theory (DFPT) [18–20].

In the following, we provide a description of the PCG algorithm implemented in `cgsolve_all`. We emphasize that this solver is not the Projected Preconditioned Conjugate Gradient (PPCG) eigensolver used for eigenvalue computations; instead, it solves in terms of bands linear systems of the form $(H - \epsilon_i + Q)x = b$ with preconditioning and projection.

2.1.3 Algorithm implemented in `cgsolve_all`

For each band $i = 1, \ldots, N_b$, the goal is to solve the linear system

$$A_i d\psi_i = d_0\psi_i, \qquad A_i \equiv H - \epsilon_i + Q. \tag{2.3}$$

The algorithm proceeds iteratively as follows.

1. Initialization. Start from an initial guess $d\psi_i^{(0)}$. Compute the residual:

$$g_i^{(0)} = A_i d\psi_i^{(0)} - d_0\psi_i, \tag{2.4}$$

[1] Mathematical formula used to find eigenvalues of a matrix, particularly in the context of Hermitian matrices.

and apply a diagonal (kinetic energy based) preconditioner:

$$h_i^{(0)} = M_i^{-1} g_i^{(0)}. \tag{2.5}$$

Define the scalar quantity $\rho_i^{(0)} = \langle h_i^{(0)}, g_i^{(0)} \rangle$.

2. Conjugate gradient iteration. For iteration $k = 0, 1, 2, \ldots$ until convergence:

1. Compute preconditioned residual norm:

$$\rho_i^{(k)} = \langle h_i^{(k)}, g_i^{(k)} \rangle,$$

   and test convergence: if $\sqrt{\rho_i^{(k)}} < \texttt{ethr}$, the band is considered converged.

2. Compute the CG update direction using:

$$\beta_i^{(k)} = \frac{\rho_i^{(k)}}{\rho_i^{(k-1)}}, \qquad h_i^{(k)} \leftarrow -h_i^{(k)} + \beta_i^{(k)} h_i^{(k-1)}.$$

3. Apply the operator A_i:

$$t_i^{(k)} = A_i h_i^{(k)}.$$

4. Determine optimal line minimization parameter:

$$\lambda_i^{(k)} = -\frac{\langle h_i^{(k)}, g_i^{(k)} \rangle}{\langle h_i^{(k)}, t_i^{(k)} \rangle}.$$

5. Update solution and residual:

$$d\psi_i^{(k+1)} = d\psi_i^{(k)} + \lambda_i^{(k)} h_i^{(k)},$$
$$g_i^{(k+1)} = g_i^{(k)} + \lambda_i^{(k)} t_i^{(k)}.$$

3. Parallel and band group synchronization. The above loop is executed independently on subsets of bands distributed across MPI band groups. At each iteration:

- convergence information is gathered from all ranks,
- global equivalence of convergence flags is checked,
- updated directions and norms are optionally reduced across the band group communicator.

The iteration stops when all bands assigned to a band group have converged.

2.1.4 Characteristics of the PCG implementation

- It is a linear solver, not an eigensolver.
- Each band solves an independent shifted system $(H - \epsilon_i + Q)x = b$.
- The preconditioner is diagonal in reciprocal space and depends on the kinetic energy.
- The projection operator Q removes components along occupied states.
- The algorithm is suitable for GPU offload and MPI band parallelization.

- Memory usage is significantly lighter than Davidson or PPCG methods.

The routine `cgsolve_all` implements a classical, bandwise preconditioned conjugate gradient (PCG) method customized for density functional perturbation theory. It provides an efficient and memory friendly procedure to solve the linear equations required by phonon perturbations, playing a central role in the performance of `ph.x`. This PCG solver is therefore a key computational kernel whose behavior and scaling properties must be understood for the optimization efforts presented in this thesis.

2.2 Dynamic Band Balancing

In the `cgsolve_all` routine of `phonon.x`, the linear systems

$$(H - \epsilon_i + Q)\, d\psi_i = d0\psi_i, \qquad i = 1, \ldots, N_{\text{bands}}$$

are distributed among MPI processes using the so–called band groups (BGs). Each BG is assigned a subset of the total number of bands and solves its own subset essentially independently.

Static distribution using `divide.f90`

The initial distribution is performed by the routine divide.f90, (`q-e/UtilXlib/divide.f90`), which receives the total number of bands `nbnd` and the communicator `inter_bgrp_comm`. The procedure computes

$$\texttt{ndiv} = \left\lfloor \frac{\texttt{nbnd}}{\texttt{nproc}} \right\rfloor, \qquad \texttt{rest} = \texttt{nbnd} \bmod \texttt{nproc},$$

assigning `ndiv+1` bands to the first `rest` ranks and `ndiv` bands to the remaining ones. Thus, the distribution is perfectly even in number of indices, but implicitly assumes that each band requires the same computational effort. This is the main point of this thesis because each band and each BG takes different computational effort.

The cost per band is heterogeneous

In practice, the computational cost associated with each band can differ significantly. This is a well-known feature of iterative diagonalisation in plane–wave DFT solvers, and arises from both physical and numerical reasons:

- **Bands near the Fermi level**. In metallic systems or systems with nearly–degenerate states, the shifted operator $(H - \epsilon_i + Q)$ can become poorly conditioned for bands whose eigenvalues lie close to the Fermi energy. Poor conditioning increases the number of CG iterations required for convergence.
- **Dispersive or strongly hybridized bands**. Bands with rapid dispersion or with mixed orbital character often lead to gradients that are harder to minimise, slowing down the CG process.
- **Orthogonality constraints**. The constraint $\langle \psi_i | S | \psi_j \rangle = 0$ for $j < i$ can be costly when two bands are nearly degenerate or share similar character. Maintaining orthogonality increases the cost of the corresponding CG iteration for those bands.
- **Preconditioning effects**. If the preconditioner does not flatten the spectrum effectively in certain energy regions, the convergence rate of CG can vary substantially from band to band.

Because of these effects, even if each BG receives the same number of bands, the *actual time to solution per BG is not uniform*. Some BGs finish earlier and wait at synchronization points such as `mp_barrier(inter_bgrp_comm)`, while others continue iterating.

Measuring the band number imbalance

As discussed earlier, the routine `divide.f90` partitions the total number of bands among the available band groups using an even, static rule. However, because the computational cost per band is intrinsically heterogeneous, it is useful to quantify how far the current band allocation deviates from an ideal uniform distribution. To this end, we compute a simple *band number imbalance* metric at each iteration.

Let N_{bands} be the total number of bands, and n_{bg} the number of band groups. The ideal number of bands per BG is

$$N_{\text{avg}} = \frac{N_{\text{bands}}}{n_{\text{bg}}}.$$

For each BG r, let N_r denote the actual number of bands assigned to that group (i.e. `my_nbnd`). The band number imbalance is then defined as the relative deviation from the ideal value:

$$\text{imbalance}(r) = 100 \times \frac{|N_r - N_{\text{avg}}|}{\max(1,\, N_{\text{avg}})}.$$

The denominator uses $\max(1, N_{\text{avg}})$ for numerical safety, even though $N_{\text{avg}} > 0$ in all realistic use cases.

Interpretation.

- $\text{imbalance}(r) = 0\%$: The BG has exactly the ideal number of bands. This corresponds to a perfectly uniform distribution.
- $\text{imbalance}(r) > 0\%$: The BG holds either more or fewer bands than the average, indicating some degree of imbalance in the static distribution.

Example. Consider the case

$$N_{\text{avg}} = 50, \qquad N_r = 60.$$

Then

$$\text{imbalance}(r) = 100 \times \frac{|60 - 50|}{50} = 20\%.$$

This means that BG r has 20% more bands than the ideal static allocation computed by `divide.f90`. Although this metric does not capture the true computational work associated with each band (which is measured through solve times), it provides a simple way to visualize how the band distribution evolves under the dynamic redistribution algorithm.

Measuring the time solve imbalance

To quantify this effect, each BG measures its local solve time using `MPI_Wtime`. Let t_r denote the local time of BG r, and

$$t_{\max} = \max_r(t_r).$$

The *waiting time* of BG r is defined as

$$t_{\text{wait}}(r) = t_{\max} - t_r.$$

Large values of $t_{\text{wait}}(r)$ indicate that the BG is idle for a significant portion of the global computation. The vector $\{t_{\text{wait}}(r)\}$ is collected through `MPI_Gather` and used to identify the fastest and slowest BGs.

Dynamic redistribution: the `cgsolve_all` new strategy

To address the imbalance, we introduce a dynamic redistribution of bands. Instead of relying on the static distribution of `divide`, we maintain a persistent vector `vec_bands_per_bg_rank(0:nproc_bg-1)` storing the current number of bands assigned to each BG.

At the end of each call to `cgsolve_all` (`cgsolve_all` is called in each iteration according to the ph_input file), the algorithm:

1. Measures the vector of waiting times $t_{\text{wait}}(r)$.
2. Identifies the slowest BG (minimum wait) and fastest BG (maximum wait).
3. Computes a relative imbalance

$$\Delta_{\text{rel}} = \frac{t_{\text{wait}}(i_{\min}) - t_{\text{wait}}(i_{\max})}{t_{\max}}. \tag{2.6}$$

4. Converts Δ_{rel} into an integer number of bands to be shifted, scaled by a learning factor β.
5. Applies limits (e.g. `QE_MAX_BAND_PROP`) to avoid over corrections.
6. Updates `vec_bands_per_bg_rank` and broadcasts it to all ranks.
7. Reconstructs the intervals `n_start:n_end` from the updated vector using prefix sums.

This dynamic strategy progressively reduces the waiting times among BGs and achieves a more balanced workload, improving the overall scalability and parallel efficiency of the linear solver in `phonon.x`.

Band redistribution strategy: the role of `adjust_bandcounts`

The redistribution of bands among band groups is performed by the routine `adjust_bandcounts`, which receives as input:

- the vector of waiting times $t_{\text{wait}}(r)$,
- the maximum solve time $t_{\max}$,
- the total number of bands N_{bands},
- the identifiers of the slowest and fastest BGs, and
- the current band count vector `vec_bands_per_bg_rank`$(0{:}n_{\text{bg}}-1)$.

The goal is to move a small number of bands from the slowest BG (the one performing more work) to the fastest BG (the one spending more time waiting). To quantify the imbalance, we use the relative difference in waiting time in eq. (2.6). If $|\Delta_{\text{rel}}|$ is below a small deadband threshold (default 1%), no redistribution is performed.

To avoid excessive corrections, the adjustment is scaled by a learning rate factor $\beta \in [0.05, 0.25]$ (default $\beta = 0.10$):

$$\Delta N_{\text{bands}} = \max\left(1,\ \text{round}\left(\beta\,|\Delta_{\text{rel}}|\,N_{\text{bands}}\right)\right).$$

This number of bands is subtracted from the slowest BG and added to the fastest BG, unless doing so would violate predefined safety limits.

Learning rate parameter β

The waiting time imbalance is typically noisy and may fluctuate from iteration to iteration due to:

- variability in CG convergence between bands,
- non-monotonic behavior of the residual norm,
- communication fluctuation (MPI timing noise),

- and variations in the Hamiltonian vector products.

If the redistribution magnitude were directly proportional to $|\Delta_{\text{rel}}|$, the system could "overshoot" and cause oscillations, where bands are moved back and forth between BGs. The learning rate β damps these oscillations and ensures stable convergence toward a balanced configuration. Values in the range $0.05 \leq \beta \leq 0.20$ provide a good compromise between responsiveness and stability. We adopted $\beta = 0.10$ as default.

Role of the upper bound `QE_MAX_BAND_PROP`

To prevent pathological distributions where one BG receives too many bands, we impose:

$$\texttt{vec\_bands}(r) \;\leq\; \texttt{QE\_MAX\_BAND\_PROP} \times N_{\text{bands}}, \qquad \text{default: } 0.60.$$

This protects the solver from:

- making the fastest BG overloaded (turning it into the new bottleneck),
- making other BGs nearly empty (which wastes memory and MPI resources),
- increasing the cost of orthogonalization for a single BG,
- and reducing numerical stability (very uneven distributions).

This upper bound ensures that even in the presence of temporary fluctuations, no BG ever receives more than a physically reasonable fraction of the total band workload.

Example: two band groups ($n_{\text{bg}} = 2$)

Consider $N_{\text{bands}} = 200$ bands initially distributed as:

$$\text{BG0} : 100, \qquad \text{BG1} : 100.$$

Suppose the measured solve times are:

$$t_0 = 12.5 \text{ s}, \qquad t_1 = 10.0 \text{ s}.$$

Thus,

$$t_{\max} = 12.5 \text{ s}, \qquad t_{\text{wait}}(0) = 0, \qquad t_{\text{wait}}(1) = 2.5.$$

BG0 is slower (minimal wait), BG1 is faster (maximal wait). The relative imbalance is:

$$\Delta_{\text{rel}} = \frac{t_{\text{wait}}(0) - t_{\text{wait}}(1)}{t_{\max}} = \frac{0 - 2.5}{12.5} = -0.20.$$

Taking $\beta = 0.10$ yields:

$$\Delta N_{\text{bands}} = \text{round}(0.10 \times 0.20 \times 200) = 4.$$

We therefore update:

$$\text{BG0} : 100 \to 96, \qquad \text{BG1} : 100 \to 104.$$

In the next iteration BG1 has more work, BG0 has less, and the waiting time difference shrinks.

Example: four band groups ($n_{\text{bg}} = 4$)

Let $N_{\text{bands}} = 240$ be initially distributed as:

$$[60, 60, 60, 60].$$

Suppose the waiting times after one CG solve are:

$$t_{\text{wait}} = [0.1,\ 1.8,\ 0.4,\ 2.3]\ \text{s},$$

so that

$$t_{\max} = 2.3\ \text{s}.$$

The slowest BG (minimal wait) is BG0, while the fastest (maximal wait) is BG3. The relative imbalance is therefore:

$$\Delta_{\text{rel}} = \frac{0.1 - 2.3}{2.3} \approx -0.96.$$

With a learning rate $\beta = 0.10$:

$$\Delta N = \text{round}(0.10 \times 0.96 \times 240) = 23.$$

Updated band distribution:

$$[37,\ 60,\ 60,\ 83].$$

This updated vector is then broadcast to all BGs, and each rank reconstructs its local interval $n_{\text{start}}{:}n_{\text{end}}$ using prefix sums. In this way the next iteration will start with this new configurations in BGs.

The parameter `QE_MAX_BAND_PROP` enforces a strict upper bound on the maximum number of bands that any band group (BG) is allowed to carry.

Given the total number of bands N_{bands}, the admissible limit is

$$N_{\max}^{(\text{BG})} = \texttt{QE\_MAX\_BAND\_PROP} \times N_{\text{bands}}.$$

For example, with `QE_MAX_BAND_PROP` $= 0.60$ and $N_{\text{bands}} = 240$, no BG may ever exceed $N_{\max}^{(\text{BG})} = 0.60 \times 240 = 144$ bands. This constraint prevents pathological redistributions where one BG becomes too heavy while others are left with very few bands.

Crucially, this bound is not applied as a multiplicative reduction to the redistribution size ΔN. If the algorithm computes a correction of, say, $\Delta N = 23$ bands, the move will be fully applied unless assigning ΔN bands to the fastest BG would exceed the limit $N_{\max}^{(\text{BG})}$. Only in that case is the adjustment rejected (or limited), ensuring that each BG remains within safe numerical and performance margins.

Considering the last example of four BGs with the initial distribution $[60,\ 60,\ 60,\ 60]$, and suppose that $\Delta N = 23$ bands should be shifted from the slowest BG to the fastest one. If the fastest BG currently holds 60 bands, the update $60 \longrightarrow 60 + 23 = 83$ is allowed, because $83 < 144$. By contrast, if the fastest BG already had 140 bands, the update $140 \longrightarrow 163$ would violate the upper bound $N_{\max}^{(\text{BG})} = 144$, and the routine `adjust_bandcounts` would skip the redistribution entirely.

This mechanism ensures that the algorithm remains stable even in the presence of large temporary imbalances or noisy timing measurements. It is therefore the combination of:

- the learning rate β,
- the deadband threshold,
- and the global upper bound `QE_MAX_BAND_PROP`, prevents extreme redistribution.

that guarantees smooth, progressive, and numerically safe convergence of the band distribution. The algorithm of the band redistribution logic implemented in `adjust_bandcounts` is shown in [1]. The routine measures the imbalance among band groups, computes a stable correction using a learning rate β, enforces upper bounds through `QE_MAX_BAND_PROP`, updates the band distribution, and broadcasts the new configuration to all MPI ranks.
Different algorithms can be used to ensure correct band redistribution between BGs, but this works very well and we do not need to modify the subroutines too much.

Algorithm 1 Dynamic band balancing implemented in `cgsolve_all`

Require: Total number of bands N_{bands}
Require: Band counts per BG: `vec_bands`$(0{:}n_{\text{bg}}-1)$
Require: Learning rate β, upper bound `QE_MAX_BAND_PROP`

Measure local solve time: $t_{\text{local}} \leftarrow$ `MPI_Wtime()`$_{\text{end}} -$ `MPI_Wtime()`$_{\text{start}}$
Compute global maximum solve time: $t_{\max} \leftarrow$ `MPI_Allreduce`$(\max t_{\text{local}})$
Compute waiting time of this BG: $t_{\text{wait}}(r) \leftarrow t_{\max} - t_{\text{local}}$
Gather all waiting times at root: $\{t_{\text{wait}}(0), \ldots, t_{\text{wait}}(n_{\text{bg}}-1)\}$
if root process **then**
    Identify slowest BG (minimal wait): $i_{\min} = \arg\min_r t_{\text{wait}}(r)$
    Identify fastest BG (maximal wait): $i_{\max} = \arg\max_r t_{\text{wait}}(r)$
    Compute relative imbalance:

$$\Delta_{\text{rel}} = \frac{t_{\text{wait}}(i_{\min}) - t_{\text{wait}}(i_{\max})}{t_{\max}}$$

    if $|\Delta_{\text{rel}}| <$ deadband **then**
        return ▷ No redistribution
    end if
    Compute suggested band shift: $\Delta N = \max\,(1,\ \text{round}(\beta\,|\Delta_{\text{rel}}|\,N_{\text{bands}}))$
    Limit by availability: $\Delta N \leftarrow \min(\Delta N,\ $ `vec_bands`$(i_{\min}))$
    if `vec_bands`$(i_{\max}) + \Delta N >$ `QE_MAX_BAND_PROP` $\cdot\, N_{\text{bands}}$ **then**
        return ▷ Avoid exceeding allowed max load
    end if
    Apply redistribution:

$$\texttt{vec\_bands}(i_{\min}) \leftarrow \texttt{vec\_bands}(i_{\min}) - \Delta N,$$
$$\texttt{vec\_bands}(i_{\max}) \leftarrow \texttt{vec\_bands}(i_{\max}) + \Delta N$$

end if
Broadcast updated vector `vec_bands` to all BG ranks
Reconstruct local indices using prefix sums: $(n_{\text{start}}, n_{\text{end}}) \leftarrow$ prefix_sum(`vec_bands`)

2.3 Performance Analysis

All performance tests were conducted on two CINECA [21] systems: the Leonardo cluster [22] and the Galileo100 cluster [2] [23]. Table {2.1} summarizes the main hardware characteristics of the computing systems used in this work.

Different parallel configurations were explored by varying the number of compute nodes, the number of MPI tasks per node, the number of OpenMP threads per task, and the number of band groups. For clarity, we adopt the compact notation:

$$[\text{nn} :: \text{MPI} :: \text{OMP} :: \text{BG}],$$

where `nn` denotes the number of nodes, `MPI` the number of MPI tasks per node and `OMP` the number of OpenMP threads per task.

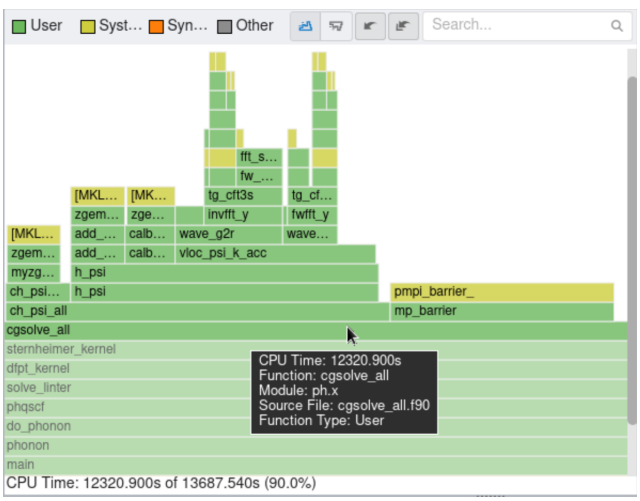
To analyze the computational behavior of `phonon.x` and identify its main performance bottlenecks, we performed a detailed profiling study using the Intel VTune Profiler [24], which is well suited for Intel-based architectures such as those of the Galileo100 cluster. The profiling results provide insight into the execution flow and the distribution of CPU time across the main routines of the code. In particular, VTune frame graphs enable a clear

[2] All tests on Galileo100 were executed using the `g100_usr_prod` QoS, corresponding to the standard production environment.

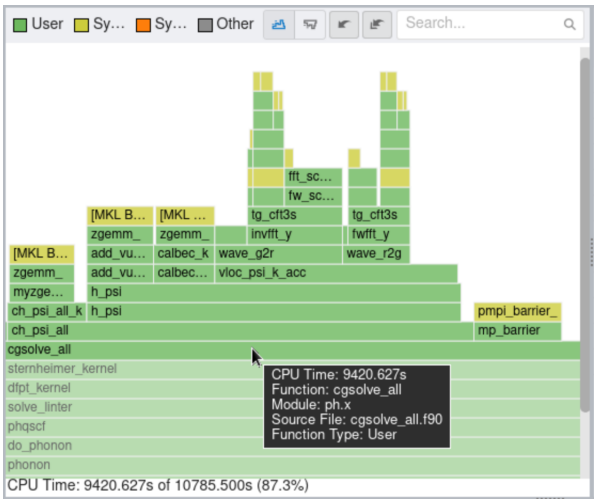
Table 2.1: Main hardware characteristics of the clusters used in this work.

Machine	**Leonardo**	**Galileo100**
Partition	Booster	g100
QoS	—	g100_usr_prod
Model	BullSequana X2135 “Da Vinci” single node GPU Blade	Dual-socket Dell PowerEdge
Nodes	3456	422
Processors	1 × Intel Xeon Platinum 8358 (Ice Lake), 32 cores, 2.6 GHz	2 × Intel Xeon Platinum 8276-8276L, 24 cores each, 2.4 GHz
Cores per node	32	48
System RAM	8×64 GB DDR4 3200 MHz (512 GB)	384 GB
Accelerators	4x NVIDIA custom Ampere A100 GPU 64GB HBM2e, NVLink 3.0 (200GB/s)	—
Network	2 x dual port HDR100 per node (400Gbps/node)	Mellanox Infiniband 100GbE

visualization of the relative cost of each routine. To assess the impact of the proposed dynamic band redistribution strategy, we compare executions of `ph.x` with and without the dynamic band-balancing mechanism implemented in `cgsolve_all.f90`. Figure 2.1 shows the corresponding VTune frame graphs. A significant reduction in total CPU time is observed, decreasing from 13687.540 s to 10785.500 s. The largest improvement is found in the `cgsolve_all` routine, whose execution time is reduced from 12320.900 s to 9420.627 s.



(a) CGSOLVE_ALL.F90 before implementing dynamical balancing. CPU Time: 12320.900s of 13687.540s (90.0%)



(b) CGSOLVE_ALL.F90 after implementing dynamical balancing. CPU Time: 9420.627s of 10785.500s (87.3%).

Figure 2.1: Frame graphs of `ph.x` obtained with VTune, using the configuration [1::24::2::2] in Galileo100 cluster.

Overall, this corresponds to an approximate reduction of 21.2% in the total CPU time and 23.5% in the CGSOLVE_ALL. As discussed previously, the magnitude of this improvement depends on the specific material system and computational configuration considered.

Figure {2.2} shows the evolution of imbalance, solution time, waiting time and the number of bands for each iteration before the implementation of band balancing. We can see that the imbalance is zero because each band group has the same number of bands. The solution time differs for each band group; here, we can see which band groups are faster and slower. The same applies to waiting time: in this case, group number 3 is not waiting because

it takes the longest to finish. Here, we can clearly see the solve time imbalance between the band groups.

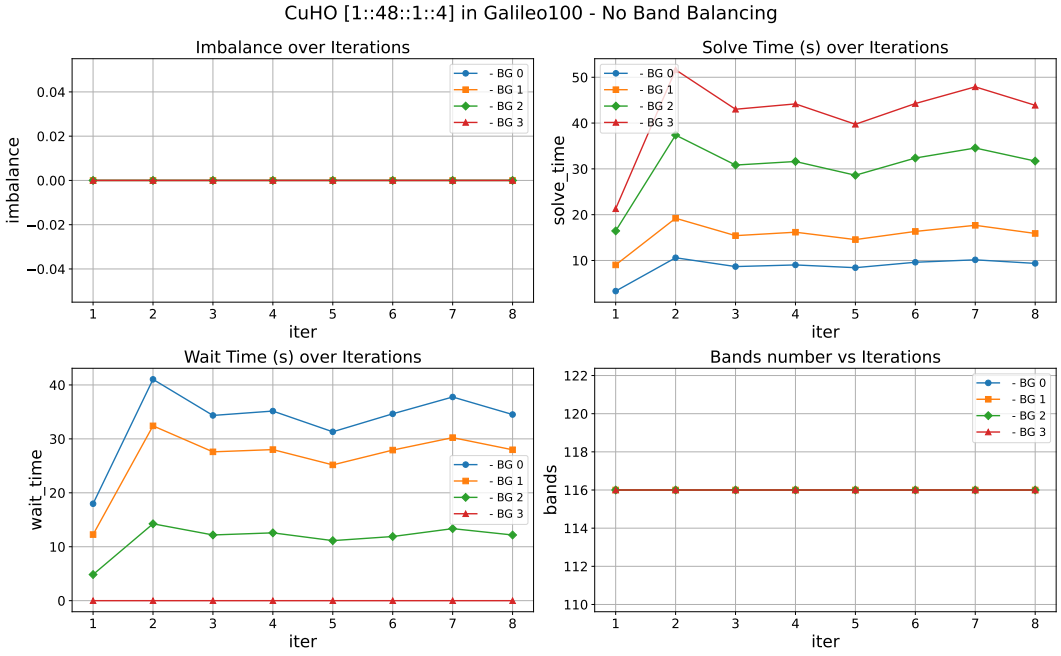


Figure 2.2: Evolution of imbalance, solution time, waiting time, and per-band behavior as a function of the number of iterations for the CuHO system on Galileo100, before the application of the dynamic band balancing strategy.

The figures {2.3,2.4} show the evolution of imbalance, solution time, waiting time and the number of bands for each iteration after the band balancing implementation for CuHO and SiO_2 systemns on Galileo100. The benchmarks exhibit markedly different behavior under the dynamic band balancing strategy, highlighting both the flexibility and the generality of the proposed method.

The CuHO system shows significant imbalance among its four bands. BG 0 is the fastest, which is why this group has more bands in the final iteration. As shown in Fig. {2.4}, both the solve time and wait time distributions differ significantly during the early iterations. This indicates that the cost per band is highly heterogeneous, likely due to band structure features such as near degeneracies or bands close to the Fermi level that require more CG iterations. The dynamic redistribution algorithm responds aggressively, moving a larger number of bands between BGs at early iterations. After several steps, the band distribution converges to an asymmetric but stable configuration that equalizes the solve times and dramatically reduces wait times.

The SiO2 benchmark (Fig. {2.4}) presents a significantly more homogeneous cost distribution from the outset. The imbalances converge to two constant numbers for BG 0 and 3, and to a different constant for BG 1 and 2. Solve times differ by only a few percent, although there is a significant difference in wait times at the beginning, they remain under 1.2 seconds throughout the process. In this case, the algorithm applies only minimal adjustments to the band distribution. The final distribution remains close to the initial $[24, 24, 24, 24]$ split, with only minor corrections such as $[30, 25, 22, 19]$. This also shows that the behavior differs for each material combination and for different selections of parameters β and `QE_MAX_BAND_PROP`. For this material, by the fifth iteration, we can see that the solve and wait times reach the minimum. However, as we want a general modification that works for most materials, we need to keep the configuration of this parameters as general as possible. This also demonstrates that the algorithm does not overcorrect when the static `divide` distribution is already close to optimal.

Taken together, these results show that dynamic band balancing adapts naturally to the underlying physics and numerical properties of each system. When the workload is uneven (CuHO), the method performs strong corrections and significant performance gains are achieved. When the workload is already homogeneous (SiO2), the algorithm makes small, stable adjustments and avoids unnecessary oscillations thanks to the learning rate β and the constraints enforced by `QE_MAX_BAND_PROP`.

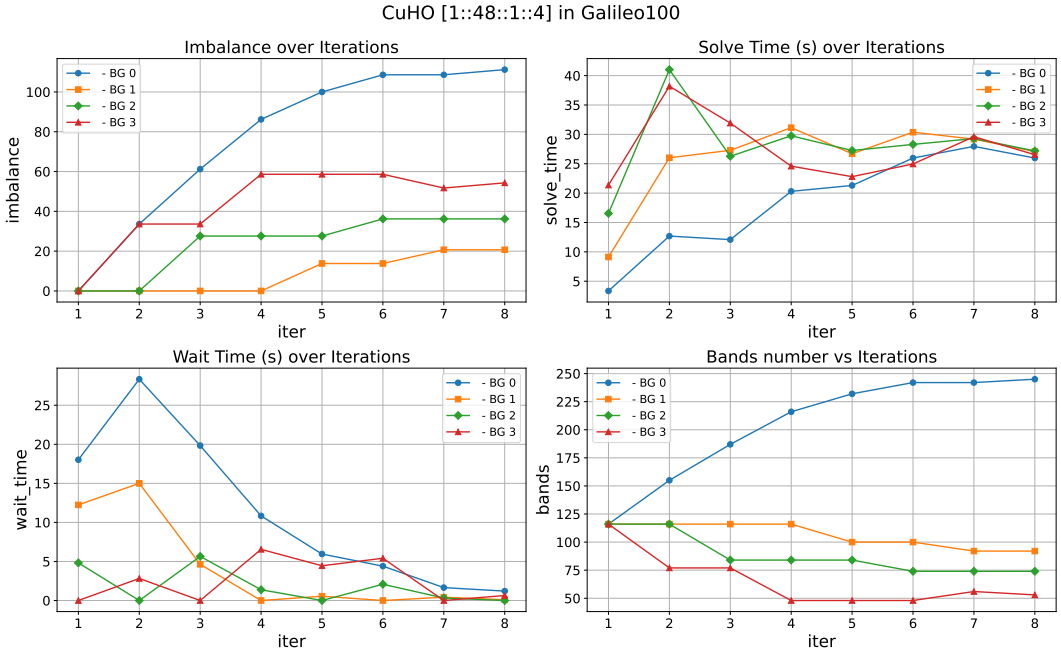


Figure 2.3: Evolution of imbalance, solution time and waiting time, as well as the number of bands, versus the number of iterations for CuHO system on Galileo100.

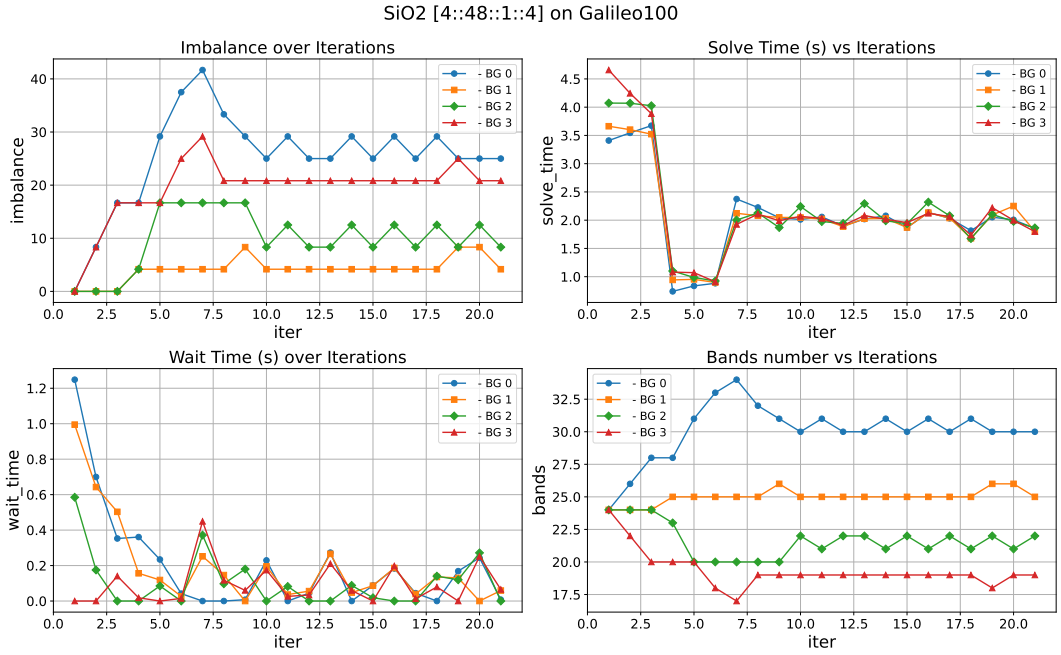


Figure 2.4: Evolution of imbalance, solution time and waiting time, as well as the number of bands, versus the number of iterations for SiO2 system on Galileo100.

In both cases, the overall effect is an increase in parallel efficiency, a reduction in synchronization overhead, and a more balanced distribution of CG iterations across band groups.

Now we evaluate the impact of the proposed dynamic band-balancing strategy on the total execution time of QE, as well as the CPU and wall times of `phonon.x`, for different material systems and parallel configurations. Unless otherwise stated, we use the notation [`nn` :: `MPI` :: `OMP` :: `BG`]. This compact format corresponds directly to the SLURM directives used in the runs.

2.3.1 Algorithmic Strong Scaling

Figure {2.5} compares the original solver and the dynamically balanced implementation for the CuHO system under three configurations: `[1::12::1::1]`, `[1::24::1::2]`, and `[1::48::1::4]` on Galileo100.

In this experiment, the physical problem size remains fixed while the degree of parallelism increases through a simultaneous growth of MPI ranks and band groups (BGs). Since BGs define the band-parallel decomposition in `PH.x`, this setup represents a strong-scaling study along the band-parallelization dimension.

Under ideal strong scaling, execution time follows $t(P) \sim 1/P$. Departures from this trend typically indicate load imbalance, communication overhead, or synchronization costs.

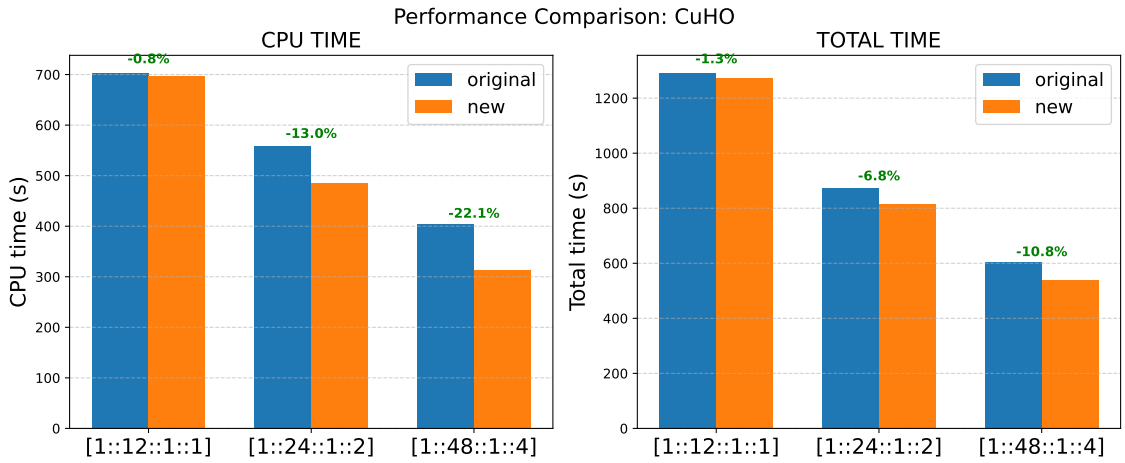


Figure 2.5: CuHO performance under increasing parallelism `[nn::MPI::OMP::BG]`. Percentage labels indicate improvements obtained with dynamic balancing.

CuHO results.

- For `BG` = 1, improvements are negligible ($\approx 1\%$). With a single band group, no inter-group imbalance exists and both solvers behave equivalently.
- For `BG` = 2, dynamic balancing reduces CPU time by 13% and total runtime by 6.8%.
- For `BG` = 4, gains increase to 22.1% (CPU) and 10.8% (total time), indicating that imbalance becomes more pronounced as band parallelism grows.

The monotonic increase in performance gains confirms that the dominant strong-scaling limitation in CuHO originates from band-level imbalance rather than hardware constraints.

SiO_2 comparison. For SiO_2 (Figure {2.6}), the node count and BG increase simultaneously: `[2::24::2::1]`, `[4::24::2::2]`, and `[8::24::2::4]` on Galileo100.

- For `BG` = 2, CPU time improves by 5.7%.

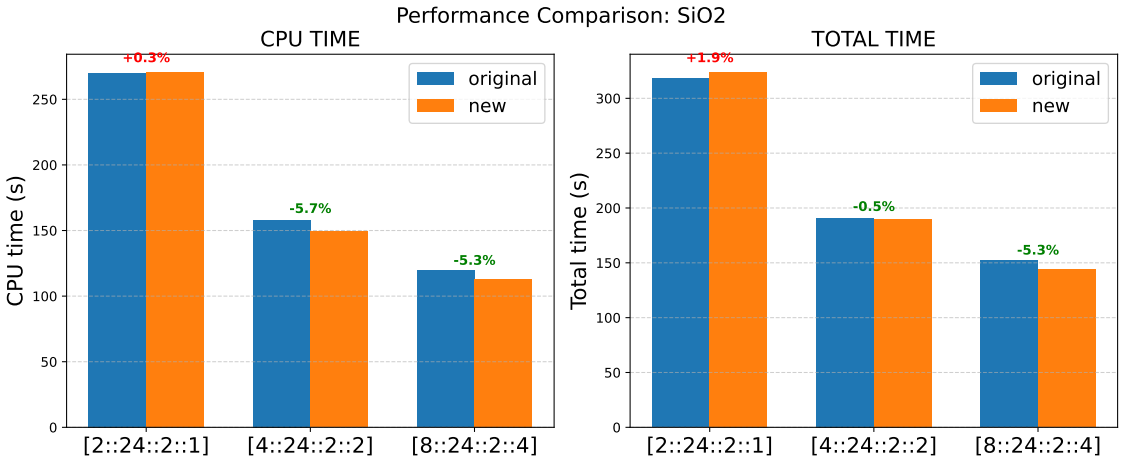


Figure 2.6: SiO_2 performance under increasing nodes and band groups.

- For BG = 4, improvements remain stable at $\approx 5\%$.

Because SiO_2 exhibits a more homogeneous band workload, the imbalance is weaker and performance gains are correspondingly smaller. Nevertheless, dynamic redistribution still yields measurable benefits once multiple BGs are introduced.

Speedup and Efficiency. To quantify scaling behavior, we compute the speedup $S(P) = t(1)/t(P)$ and the efficiency $Eff = S(P)/P$ using the CuHO data. Results are shown in Figure {2.7}.

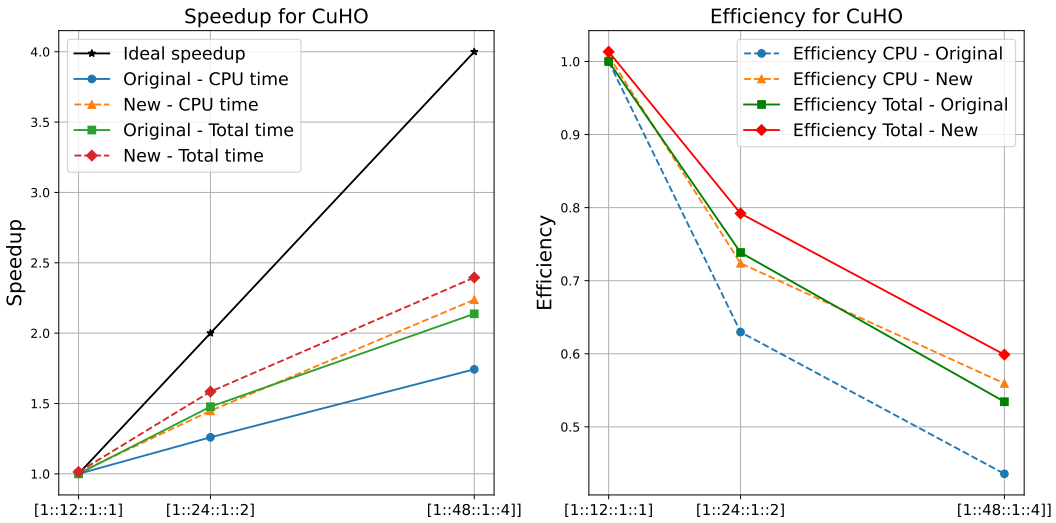


Figure 2.7: Speedup and efficiency for CuHO under band strong scaling.

While ideal linear speedup is unattainable for realistic applications, the dynamically balanced solver consistently approaches the ideal trend more closely than the original implementation. This improvement reflects reduced waiting times and improved parallel efficiency at higher BG counts.

2.3.2 Impact of Band-Group Decomposition under Fixed Parallel Resources

We analyze an additional experiment in which the number of band groups is varied while the computational resources remain strictly constant, see figure {2.8}. All simulations are executed on a single node using 32 MPI ranks and one OpenMP thread per rank, with MPI layouts `[1::32::1::1]`, `[1::32::1::2]`, and `[1::32::1::4]` on Leonardo.

Because neither the number of nodes nor the total MPI ranks change, this study isolates the algorithmic impact of band-group decomposition. Only the distribution of electronic bands across MPI ranks is modified, while the physical workload and hardware allocation remain unchanged. Performance variations can therefore be attributed exclusively to workload partitioning and synchronization effects inside the conjugate-gradient solver.

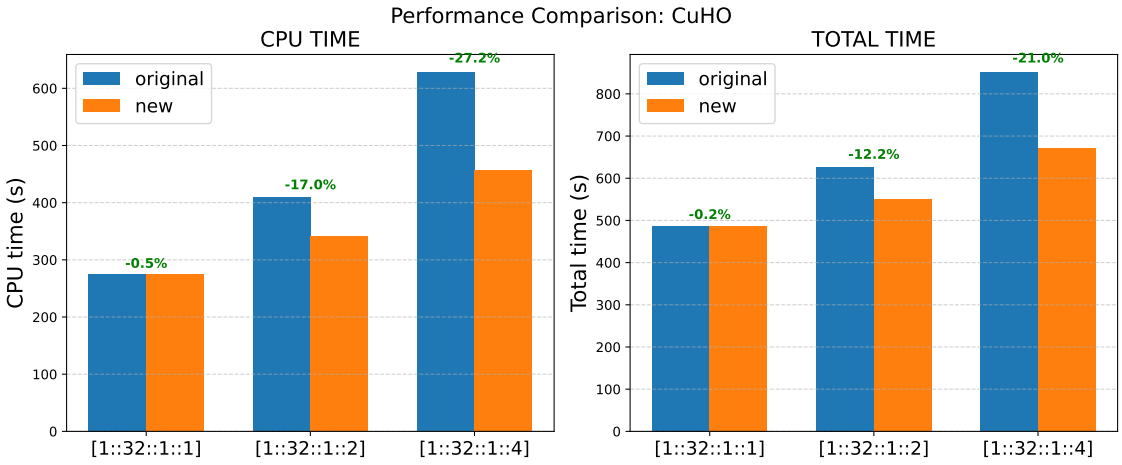


Figure 2.8: Performance comparison for CuHO under fixed computational resources (1 node, 32 MPI ranks, 1 OpenMP thread) while varying the number of band groups.

Observations.

- For BG = 2, dynamic balancing reduces CPU time by 17% and total runtime by 12.2%. The introduction of multiple BGs reveals heterogeneous band costs that remain hidden under a single-group configuration.
- For BG = 4, improvements increase to 27.2% in CPU time and 21.0% in total runtime. The original solver exhibits pronounced imbalance and waiting time between BGs, whereas the dynamic implementation restores a more uniform workload distribution.

Interpretation.

Increasing the number of BGs refines band parallelism by assigning fewer bands to each group. While finer decomposition increases potential concurrency, it also magnifies workload heterogeneity across groups. In the static solver, this results in idle periods and synchronization delays that limit efficiency.

The dynamically balanced solver compensates for these effects by adapting the band allocation at runtime, reducing waiting times and improving resource utilization. The monotonic growth of performance gains with increasing BG confirms that band imbalance is the dominant bottleneck in fine-grained band parallel executions for the CuHO system.

This experiment therefore demonstrates that scalability limitations in band-parallel runs originate primarily from algorithmic imbalance rather than hardware constraints, and that dynamic redistribution effectively restores efficiency under refined band decomposition.

2.3.3 Cross material interpretation

The combined results of CuHO and SiO2 reveal two general principles:

1. Dynamic balancing is most beneficial when the physical system exhibits heterogeneous band convergence behavior. This is the case for CuHO, where workload differences among BGs are large.
2. Systems with uniform band behavior (SiO2) exhibit smaller but still non-zero improvement when BG ≥ 2 (remember that for BG=1 we don't make any changes because there is only one BG). Even mild imbalance is corrected by the redistribution algorithm.

3. Increasing the number of BGs amplifies the benefit. Both materials demonstrate that improvements increase with the number of BGs, as static partitioning becomes progressively less effective. However, most users only use two or four BGs.

4. The dynamic method consistently decreases CPU time, total time, and wall time, without introducing instability. This validates the design choices of using a learning rate β and respecting the upper bound `QE_MAX_BAND_PROP`.

Overall, these results demonstrate that the proposed dynamic band balancing strategy systematically improves the parallel efficiency of `cgsolve_all` and `phonon.x` across different materials and parallel configurations.

2.3.4 Dynamical balancing with GPUs

Figure {2.9} illustrates the evolution of solution time, wait time, wait ratio, and band distribution as a function of the number of iterations for the CuHO system, comparing CPU and GPU executions on Leonardo.

The solution time and wait time are normalized by their maximum values:

$$t_{\text{norm}}(i) = \frac{t(i)}{\max(t)},$$

while the wait ratio provides a relative measure of idle time:

$$\text{wait\_ratio} = \frac{t_{\text{wait}}}{t_{\text{solve}} + t_{\text{wait}}}.$$

A similar convergence behavior is observed for both CPU and GPU executions: the waiting time decreases towards negligible values, and the solution time becomes more uniform across band groups. This indicates that the dynamic band balancing strategy behaves consistently across architectures.

From a performance perspective, although the GPU execution is overall faster, the impact of the proposed method in GPU executions can be understood through the interaction between CPU tasks and GPU devices. In the current implementation, each MPI rank (associated with a band group) is responsible for preparing and offloading work to a specific GPU. As a result, load imbalance at the CPU level directly translates into an unbalanced workload across GPUs.

In the presence of imbalance, some ranks complete their assigned work earlier, causing their corresponding GPUs to remain idle while waiting for the slowest ranks to reach synchronization points. This leads to reduced device utilization and degraded parallel efficiency.

By dynamically redistributing bands across band groups, the proposed method ensures that each MPI rank processes a similar amount of work. Consequently, the workload offloaded to each GPU becomes more uniform, reducing idle time across devices and improving overall efficiency.

This behavior is confirmed by the normalized timing results in Figure 2.9, where both CPU and GPU executions exhibit the same convergence pattern: the waiting time approaches zero and the solution time becomes uniform across band groups. This consistency indicates that the workload is effectively balanced at both the CPU and GPU levels.

Overall, although the optimization is applied at the CPU level, it induces a balanced execution on GPUs, improving device utilization and contributing to a more efficient time-to-solution.

2.4 Extension to Multi-Node Configurations

The performance analysis presented so far has primarily focused on single-node configurations. This choice is motivated by two main reasons. First, many production workloads in Quantum ESPRESSO are executed on a limited number of nodes due to resource constraints. Second, restricting the analysis to a single node allows us to

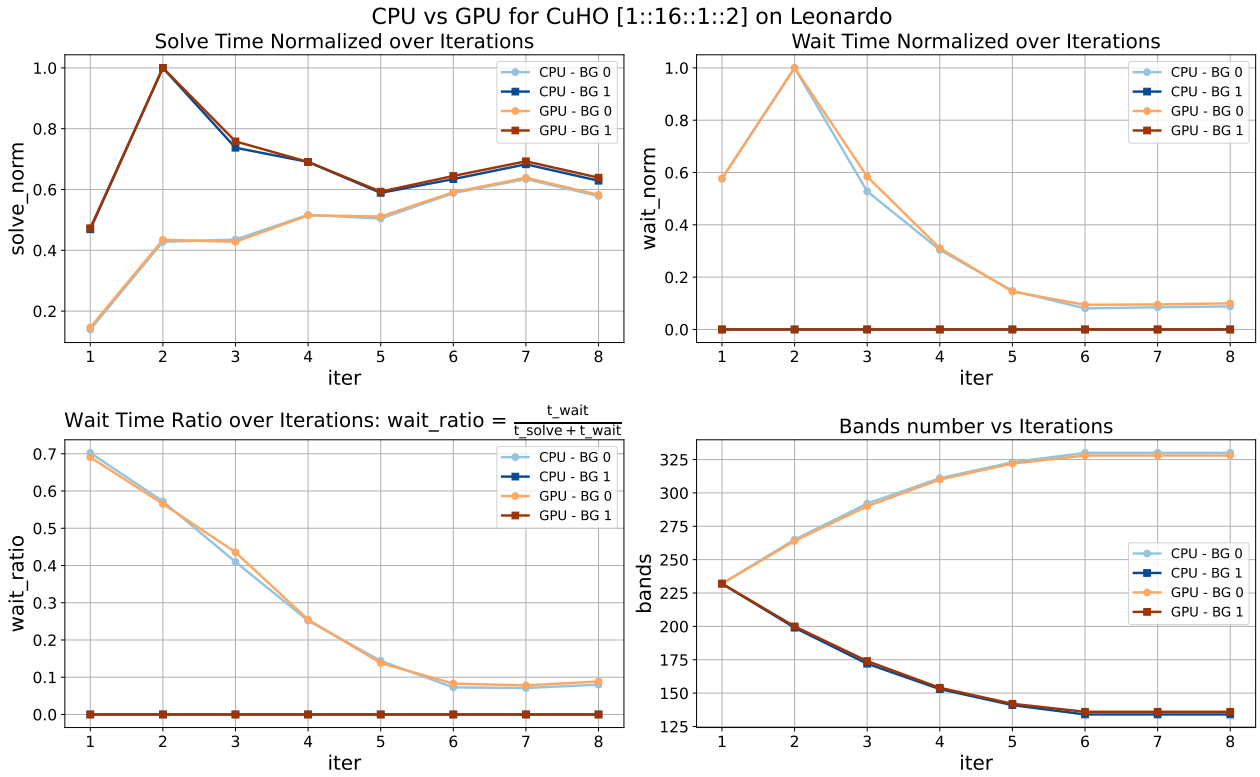


Figure 2.9: Evolution of imbalance, solution time, wait time and the number of bands versus the number of iterations for CuHO system comparing CPU vs GPU on Leonardo.

isolate the intrinsic load imbalance of the band-parallel solver, avoiding additional effects introduced by inter-node communication.

To assess the robustness of the proposed dynamic band balancing strategy, we extend the analysis to multi-node configurations on the Galileo100 system. Additional tests were performed for different materials and parallel layouts, including LaO and SiO_2.

Figure {2.10} shows the behavior for the LaO system using the configuration `[4::48::1::4]`. As the iterations progress, the waiting time rapidly decreases and approaches zero, while the solution time converges to similar values across all band groups. This indicates that the load imbalance is effectively mitigated by the dynamic redistribution.

A similar behavior is observed for a different parallel configuration of the same system. Figure {2.11} shows results for `[8::24::2::4]`, where again the waiting time is progressively reduced and the solution time becomes nearly uniform across band groups.

The same trend is confirmed for a different material system. As shown in Figure {2.12}, for SiO_2 with configuration `[4::24::2::2]`, the waiting time quickly decreases to negligible values, while the solution time stabilizes across band groups, demonstrating consistent convergence of the balancing procedure.

We also explored different values of the tuning parameter β (e.g., $\beta = 0.05$ and $\beta = 0.10$), observing only minor differences in the convergence behavior. This suggests that the method is robust with respect to this parameter, and a fixed value can be used in practice without loss of performance.

From a scalability perspective, band parallelization (BG) increases the available concurrency by distributing electronic states across multiple band groups. However, as the number of BGs grows, load imbalance between groups can degrade parallel efficiency. The proposed dynamic band balancing strategy mitigates this effect by redistributing bands at runtime, reducing idle time and improving workload uniformity across band groups. Consequently, it enhances parallel efficiency and is expected to remain effective at larger scales. This behavior suggests that the proposed method enables scalable band-parallel execution by minimizing idle time within band groups.

Even in large-scale simulations where real-space and reciprocal-space (R&G) decomposition is used to distribute

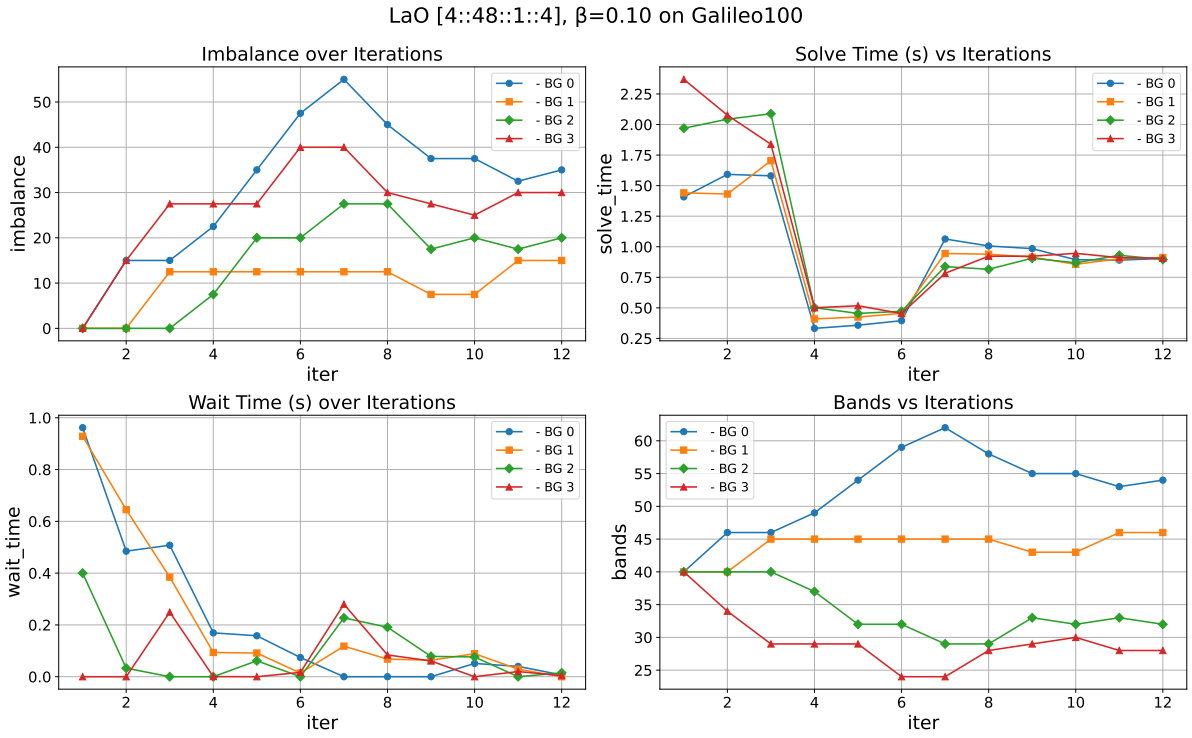


Figure 2.10: Evolution of load imbalance, solution time, waiting time, and band distribution as a function of iterations for the LaO system on Galileo100 using configuration `[4::48::1::4]`.

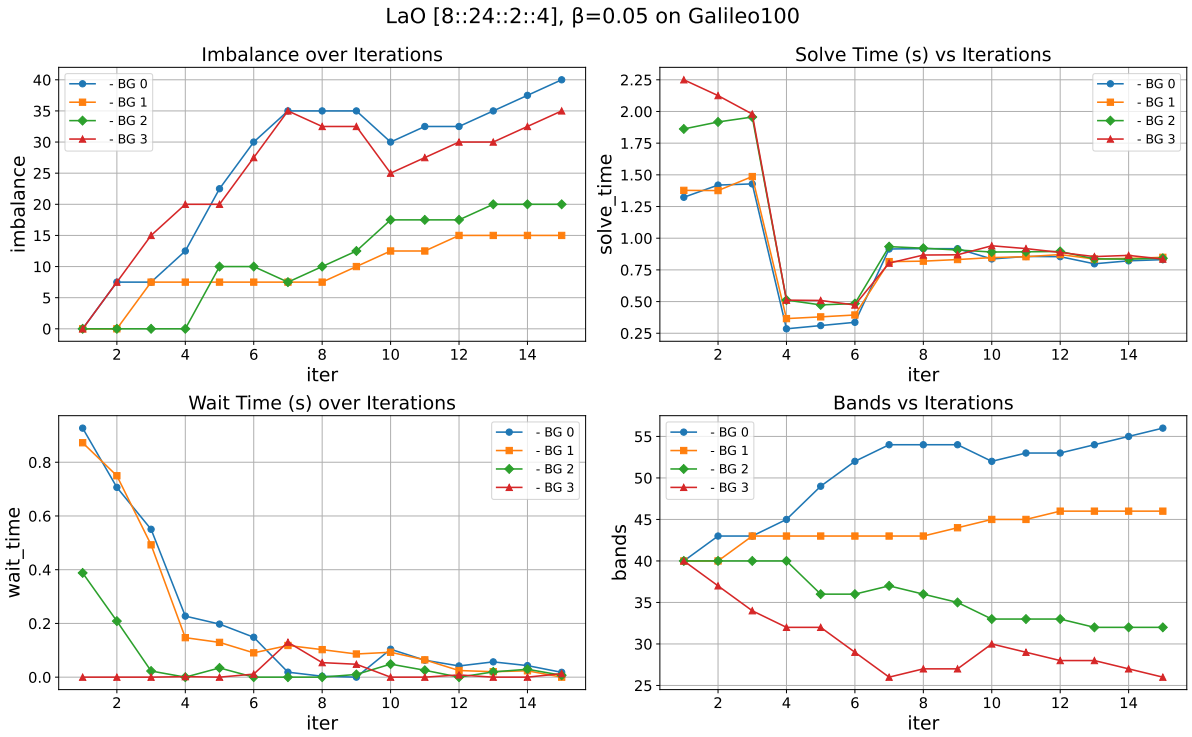


Figure 2.11: Evolution of load imbalance, solution time, waiting time, and band distribution as a function of iterations for the LaO system on Galileo100 using configuration `[8::24::2::4]`.

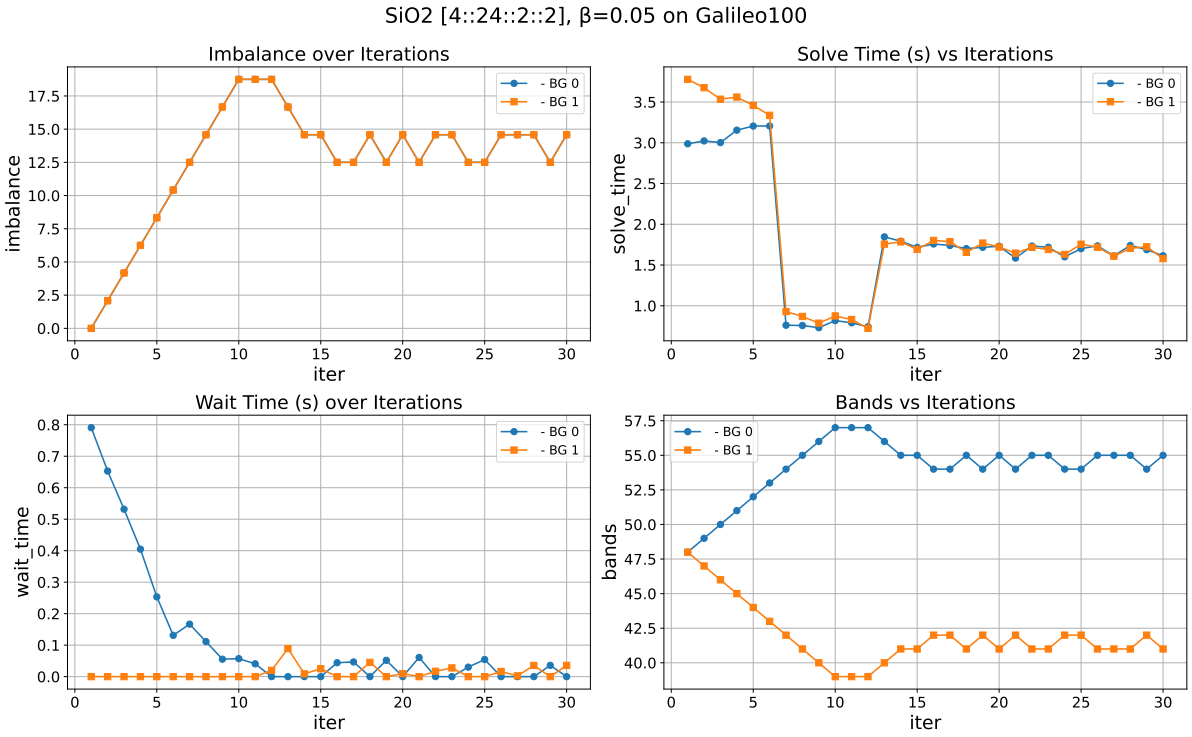


Figure 2.12: Evolution of load imbalance, solution time, waiting time, and band distribution as a function of iterations for the SiO_2 system on Galileo100 using configuration `[4::24::2::2]`.

the workload across multiple nodes, band parallelization (BG) provides an additional level of concurrency. The proposed dynamic band balancing strategy operates at this level, improving workload distribution across band groups and reducing idle time. Therefore, it is expected to complement the existing R&G parallelization and remain effective at larger scales.

Overall, these results demonstrate that the dynamic band balancing strategy is not limited to single-node scenarios, but remains effective and robust in multi-node configurations.

2.5 Impact of the Band-Balancing Strategy Beyond `PH.x`

Although the performance analysis presented in this work focuses primarily on the `PH.x` executable, it is important to emphasize that the proposed dynamic band-balancing strategy is not restricted to phonon calculations. The optimization was implemented directly within the routine `cgsolve_all`, located in the `LR_Modules` library of Quantum ESPRESSO.

The `cgsolve_all` routine constitutes the core conjugate-gradient solver employed in linear-response calculations based on the Sternheimer equation. Since this solver is compiled as part of the shared library `libqe_lr_modules`, it is linked by several executables across the Quantum ESPRESSO ecosystem. Consequently, any improvement introduced at this level propagates automatically to all applications relying on the same linear-response infrastructure.

Code inspection and build-system analysis show that, in addition to `PH.x`, the `cgsolve_all` solver is used by multiple linear-response and excited-state executables, including `EPW.x`, `alpha2f.x`, `HP.x`, TDDFPT solvers (e.g., `turbo_lanczos.x`, `turbo_davidson.x`, `turbo_eels.x`), as well as many-body and Koopmans-compliant workflows such as `pw4gww.x` and `kcw.x`. These executables solve linear systems of the same mathematical structure, where the workload is distributed over electronic bands and therefore subject to the same load-imbalance mechanisms addressed in this work.

As a result, the dynamic redistribution of bands among band groups introduced here improves load balance not only for phonon calculations, but more generally for the entire class of linear-response simulations implemented in

Quantum ESPRESSO. The performance gains demonstrated for `PH.x` can therefore be interpreted as representative of a broader improvement affecting multiple scientific workflows relying on iterative conjugate-gradient solvers.

This observation highlights that the proposed optimization operates at the algorithmic infrastructure level rather than at the application level, making it broadly applicable across different physical modules without requiring further modifications.

2.6 Conclusions and Summary

This work introduced and evaluated a dynamic band balancing strategy for the `cgsolve_all` routine in `phonon.x`, aimed at mitigating the load imbalance inherent to the static band partition implemented in `divide.f90`. The analysis was carried out across multiple material systems (CuHO and SiO_2) and diverse MPI and node configurations.

The results consistently demonstrate that static band partitioning is not sufficient for modern parallel architectures. The even split of bands neglects substantial variations in computational cost arising from band dependent CG convergence rates, near degeneracies, and system specific electronic structure features. As a consequence, significant inter group waiting times emerge, especially as the number of band groups (BGs) increases.

The main findings of this study can be summarized as follows:

1. **Band workload heterogeneity is strongly system dependent.** CuHO exhibits pronounced imbalance due to variations in CG convergence behavior across bands, whereas SiO_2 presents a more homogeneous workload distribution. This confirms that imbalance originates from physical and numerical properties of the system, not from the parallel layout itself.

2. **Static band distribution becomes increasingly suboptimal as BG increases.** For `BG` ≥ 2, the original solver shows measurable and growing imbalance, which scales with the number of band groups. The artificial even partition fails to reflect the true per band computational cost.

3. **Dynamic redistribution significantly improves parallel efficiency.** By reallocating bands according to measured solve times, the new solver reduces inter group waiting times and equalizes solve times across BGs. CPU and total wall time improvements range from **5–27%**, exceeding **20%** in strongly imbalanced cases such as CuHO.

4. **The algorithm is adaptive, stable, and convergent.** The learning rate β prevents over corrections, while `QE_MAX_BAND_PROP` limits excessive redistribution. No oscillatory or pathological behavior was observed. The band allocation stabilizes after a few iterations, converging to a configuration that reflects the actual computational load.

5. **The method is robust across materials and architectures.** The strategy performs well when scaling BG via additional MPI ranks per node (CuHO tests) and when increasing node count (SiO_2 tests). The improvement is therefore not tied to a specific system size, MPI layout, or hardware configuration.

6. **The final band distribution approximates an optimal partition.** After convergence, each BG receives a number of bands proportional to its effective per band workload, achieving a near optimal balance that cannot be predicted a priori by any static scheme.

Overall, the dynamic band balancing mechanism constitutes a robust and portable enhancement of the `cgsolve_all` solver. It systematically reduces load imbalance, improves parallel efficiency, and enhances scalability of large-scale phonon calculations in Quantum ESPRESSO.

Importantly, the proposed optimization is implemented at the level of the shared linear-response infrastructure, namely within the `LR_Modules` library where the `cgsolve_all` routine is defined. As a consequence, its impact is not limited to the `PH.x` executable analyzed in this work. Since the same conjugate gradient solver is employed by several Quantum ESPRESSO linear response and excited state applications including electron-phonon, TDDFPT,

many-body perturbation, and Koopmans compliant workflows the improvements introduced here are expected to propagate transparently to multiple executables without additional code modifications.

Therefore, the performance gains observed for phonon calculations should be interpreted as representative of a broader scalability improvement affecting a significant fraction of Quantum ESPRESSO simulations relying on iterative Sternheimer-based solvers.

While the dynamic band-balancing strategy substantially reduces solver-level waiting times by adapting the band distribution to the measured runtime cost, the total time per band group is not always a simple linear function of the number of assigned bands. In particular, the application of the Hamiltonian in plane-wave DFT involves non-local projectors, FFT-based real/reciprocal transforms, and collective communications whose performance can introduce structural synchronization effects. For this reason, in the next chapter we move from solver-level timing to a kernel-level perspective and analyze the projector pathways in `pw.x`, comparing reciprocal-space and real-space implementations to identify additional sources of imbalance beyond band partitioning.

Chapter 3

Projectors in Quantum ESPRESSO

In plane-wave pseudopotential methods, the non-local part of the ionic potential is applied through separable projectors, whose evaluation becomes a recurring kernel in both ground-state (`pw.x`) and linear-response workflows. Although band-parallel strategies can reduce solver-level imbalance, the end-to-end performance may still be limited by projector-related routines and their interaction with FFT-based grid transformations and MPI collectives. In this chapter we analyze the two main projector evaluation pathways available in Quantum ESPRESSO, the reciprocal-space and the real-space projector formulations, and we relate their computational structure to the waiting-time patterns observed in our timing measurements. The goal is to establish a clear baseline for the well-balanced reciprocal-space path and to identify the structural origin of the imbalance emerging in the real-space path under the same hardware configuration.

3.1 General Framework of the Plane-Wave Method

Quantum ESPRESSO implements the plane-wave pseudopotential framework (PW + PP) as a practical strategy to solve the Kohn-Sham Hamiltonian equations efficiently. In this approach, the Kohn–Sham orbitals are expanded in a plane-wave basis:

$$\psi_{n,\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{G}} c_{n,\mathbf{k}}(\mathbf{G}) e^{i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}} \tag{3.1}$$

Due to crystal periodicity, only reciprocal lattice vectors contribute to the expansion. As a consequence, the electronic problem is naturally formulated in reciprocal space, where the kinetic operator is diagonal and the Kohn–Sham equation becomes an algebraic eigenvalue problem for the coefficients $c_{n,\mathbf{k}}(\mathbf{G})$.

The number of plane waves is controlled by a kinetic energy cutoff E_{cut}, which defines a sphere in reciprocal space. Only $\mathbf{G}$-vectors satisfying

$$\frac{|\mathbf{k}+\mathbf{G}|^2}{2} < E_{\text{cut}} \tag{3.2}$$

are included. The electronic density, being quadratic in the orbitals, requires Fourier components up to approximately twice this cutoff, leading to a larger reciprocal-space sphere for density-related quantities.

3.2 Grid Representation and FFT Usage

In practice, reciprocal-space quantities are embedded into a regular three-dimensional grid compatible with FFT algorithms. Fourier components outside the cutoff sphere are set to zero. The Fast Fourier Transform (FFT) enables efficient transitions between:

- **Reciprocal space:** where the kinetic operator and non-local pseudopotentials are treated efficiently.

- **Real space:** where multiplication by the local potential and density construction are computationally cheaper.

The Hamiltonian is never assembled explicitly as a dense matrix. Instead, its action on an orbital is implemented as a sequence of reciprocal-space operations, real-space multiplications, and forward/backward FFT transformations. This hybrid real/reciprocal structure is intrinsic to `pw.x` and fundamentally determines its computational profile.

3.3 Distributed FFTs and Parallel Decomposition

In parallel executions, the three-dimensional FFT grid is distributed among MPI ranks. As described in detail in [25], two main decompositions are employed in Quantum ESPRESSO:

- **Stick decomposition in reciprocal space:** only Fourier components inside the cutoff sphere are stored. These components are arranged in lines ("sticks") parallel to a given direction and distributed among MPI ranks to balance workload.
- **Slab decomposition in real space:** each rank holds full two-dimensional planes (e.g., xy planes) for a subset of grid points along the third direction.

A three-dimensional FFT proceeds in stages:

1. Local one-dimensional FFTs are performed along the stick direction.
2. A global `all-to-all` communication redistributes data from stick to slab layout.
3. Additional FFTs are performed locally in the new layout.

The `all-to-all` communication step is particularly critical. It reshapes the data distribution across processes and often represents a dominant scalability bottleneck, especially in strong-scaling regimes. Its performance depends on message size, network latency, effective bandwidth, and the ability to overlap communication and computation.

3.4 Batching and Communication Computation Overlap

Typical FFT grid sizes in realistic simulations are not large enough to saturate network bandwidth when a single FFT is distributed across many nodes. However, `pw.x` applies FFTs to many Kohn–Sham orbitals.

This allows grouping orbitals into batches, increasing the effective communication volume per collective operation. As discussed in [25], further subdividing batches into sub-batches enables overlapping of:

- Local FFT computations,
- Device memory transfers (in GPU runs),
- `all-to-all` communications.

Proper tuning of batch and subbatch sizes is essential to reach bandwidth saturation regimes and avoid latency dominated performance. Otherwise, scaling degradation may appear even if arithmetic workload per rank seems balanced.

3.5 Central Role of Reciprocal Space

From an algorithmic standpoint, reciprocal space is the natural representation of the plane wave method:

- The kinetic operator is diagonal in **G**-space.
- Non-local pseudopotential projectors are formulated in reciprocal space.
- Only reciprocal lattice vectors consistent with periodic boundary conditions are physically meaningful.

Although certain operations are more efficient in real space, the mathematical structure of the method is fundamentally based in reciprocal space.

3.6 Connection to the Present Work and Timing Analysis

The goal of the present study is not merely to redistribute bands in order to reduce imbalance. In this case the observed performance degradation may not be solely due to uneven band partitioning, but rather to structural aspects of the algorithm.

The application of the Hamiltonian in `pw.x` repeatedly involves:

- Evaluation of non-local projectors in reciprocal space,
- Forward and inverse distributed FFTs,
- Global `all-to-all` communications,
- MPI reductions and synchronization points.

Therefore, the total execution time per band group cannot be modeled simply as proportional to the number of assigned bands. Instead, it depends on a more complex functional relationship involving:

- Number of bands,
- Number of plane waves (N_{PW}),
- Number of projectors (N_{kb}),
- MPI grid decomposition,
- Communication patterns in distributed FFTs.

The timings we are measuring (e.g., solve time, waiting time t_{wait}, and kernel-specific timings) must be interpreted within this global algorithmic structure.

If a given band group exhibits larger waiting times, this may originate from:

- Variations in effective projector workload,
- Non-linear effects in `all-to-all` communications,
- Interaction between band parallelization and FFT grid decomposition,
- Incomplete bandwidth saturation or communication imbalance.

Consequently, before concluding that the imbalance is purely a band-distribution issue, it is necessary to understand how the reciprocal-space formulation, distributed FFTs, and collective communications jointly determine the observed temporal behavior.

This theoretical and algorithmic background provides the foundation for interpreting our profiling results.

3.7 Fine-Grained Timing of Projector Kernels

In order to understand whether the observed imbalance originates purely from band partitioning or from structural properties of the non-local projector path, we instrument the main routines responsible for projector evaluation and accumulation, namely those implemented in `Modules/becmod.f90` especially the subroutine `calbec_gamma_acc` and `PW/src/compute_becsum.f90` especially the subroutine `sum_bec`.

Rather than measuring only high-level solver timings, we collect fine-grained performance metrics inside the projector kernels. This allows us to separate dense linear algebra costs, communication costs, and synchronization-induced waiting times. As we can assume the imbalance may not be explained solely by the number of bands assigned to each group; therefore, a kernel-level analysis is required.

For each invocation of the selected routines, we write timing records.
The timing-related quantities are:

- $t_{\mathbf{gemm}}$: Time spent in dense linear algebra kernels (typically DGEMM-like contractions corresponding to projector–orbital overlaps).
- $t_{\mathbf{mpsum}}$ (logged as `t_comm`): Time spent in MPI communication, typically reductions or collective operations required to accumulate projector contributions across ranks.
- $t_{\mathbf{tot}}$: Total time spent in the instrumented region for the given call and rank.
- $t_{\mathbf{avg}}$: Average time across all ranks participating in the communicator.
- $t_{\mathbf{wait}}$: Waiting time, defined as the difference between the maximum time across ranks and the local time. This measures synchronization-induced slack due to imbalance.
- **imb**: Imbalance metric, typically defined as a normalized measure of dispersion (e.g., relative deviation from the average or from the maximum), $imb = t_{max}/t_{avg}$.

3.7.1 Interpretation Strategy

This decomposition enables us to distinguish between different performance regimes:

- If t_{gemm} dominates and scales proportionally to the local band count, the kernel is compute-bound and imbalance may be strongly correlated with band distribution.
- If t_{mpsum} dominates, the kernel is communication-bound, and imbalance may arise from collective synchronization effects rather than arithmetic workload.
- If t_{wait} is significant even when t_{gemm} and t_{mpsum} appear balanced, this suggests structural asymmetries in the interaction between band parallelization and FFT/grid decomposition.

In particular, since projector evaluation scales as $O(N_{PW} \cdot N_{kb} \cdot kdim)$, while communication costs depend on collective patterns and buffer sizes, the total time cannot be modeled as a simple linear function of the number of bands.

By correlating $(npw, nkb, kdim)$ with $(t_{\text{gemm}}, t_{\text{mpsum}}, t_{\text{wait}})$, we can determine whether the observed imbalance is:

1. Purely band-count driven,
2. Communication-induced,
3. Or structurally rooted in the non-local projector path.

This fine-grained timing methodology provides the foundation for the performance analysis presented in the next section, where we compare reciprocal-space and real-space projector paths and assess the origin of the observed waiting times.

3.8 Reciprocal-Space Projector Path: Baseline Behavior

In order to establish a reference performance model, we first analyze the reciprocal-space projector path under a controlled configuration of 1 MPI rank per GPU on Leonardo [22] using the Booster partition described in table {2.1}.

This configuration eliminates intra-GPU MPI contention and allows us to observe the intrinsic behavior of the algorithm without artificial communication amplification. The goal is to determine whether the observed solver imbalance originates from projector arithmetic, band partitioning, or structural communication effects.

3.8.1 Instrumented Kernels

The reciprocal-space projector evaluation is primarily performed inside:

- `Modules/becmod.f90`: `calbec_gamma_acc`
- `PW/src/compute_becsum.f90`: `sum_bec`

For each call we measure:

- Compute time (t_{gemm}),
- Communication time (t_{comm}),
- Total time (t_{tot}),
- Waiting time (t_{wait}),
- Imbalance metric.

Rather than analyzing every low-level variable, we focus on the waiting time t_{wait}, since it directly captures synchronization-induced slack and therefore reveals imbalance between ranks.

3.8.2 Observed Behavior (1 Rank per GPU)

Figures {3.1} and {3.2} show the timing results for `calbec_gamma` and `compute_becsum` with 1 rank per GPU.

The main observations are:

- The compute contribution (t_{gemm}) is nearly identical across ranks.
- Communication remains moderate and does not dominate.
- The waiting time t_{wait} is small and stable.
- The imbalance metric fluctuates close to unity.

The first call by rank usually includes “cold start” (initial planning, cold hidden tasks, assignments, initial communications/FFT plans, etc.) then we can consider this call as an atypical value, and in this way focus on the steady state performance after the first calls.

These results indicate that, under this configuration, the reciprocal-space projector path is well balanced. The workload per rank scales consistently with the number of bands, and no systematic synchronization penalty is observed.

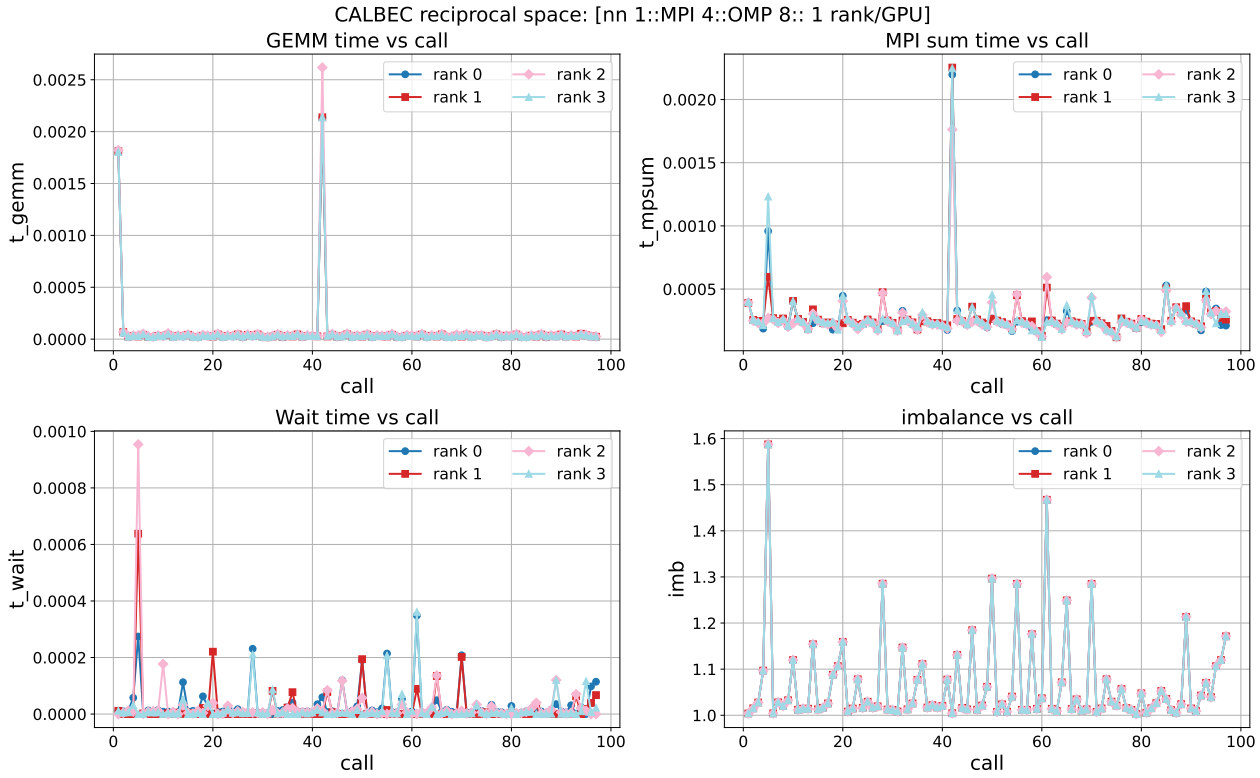


Figure 3.1: `calbec_gamma` reciprocal space for 1 rank per GPU.

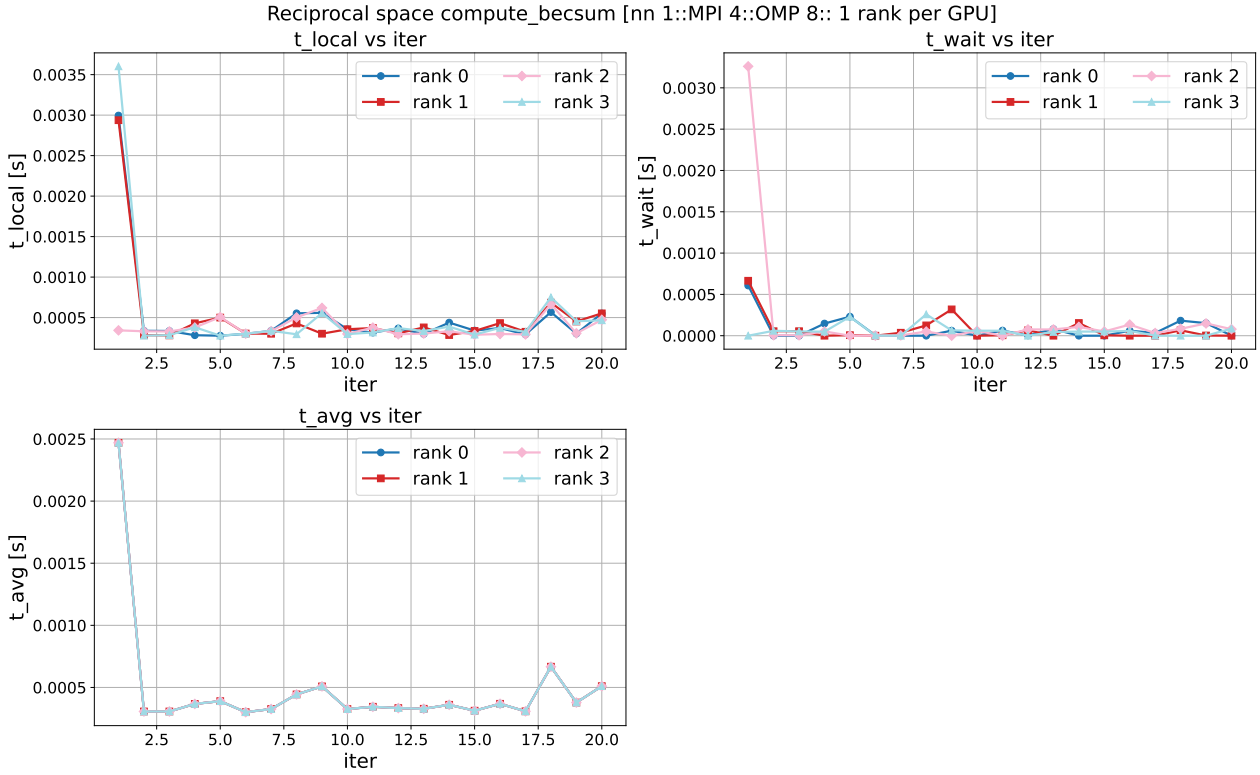


Figure 3.2: `compute_becsum` time series, 4 MPI ranks (1 rank/GPU).

3.8.3 Interpretation

From an algorithmic standpoint, this behavior is expected.

In reciprocal space:

- The kinetic operator is diagonal.
- Projector contractions are dense linear algebra operations.
- The stick decomposition distributes plane waves uniformly.
- Collective reductions are limited and predictable.

As a consequence, the total time per band group can be approximated as:

$$T_{\text{reciprocal}} \approx T_{\text{compute}}(N_{PW}, N_{kb}, kdim) + T_{\text{comm}}, \tag{3.3}$$

with T_{compute} dominating and scaling smoothly with the local band count.

Under 1 rank per GPU, communication overhead does not introduce significant waiting time, and therefore the solver-level imbalance cannot be attributed to reciprocal-space projectors.

This reciprocal-space behavior provides a baseline: if imbalance appears elsewhere, its origin must be sought in a different part of the algorithmic path.

3.9 Real-Space Projector Path

We now analyze the real-space projector path under the same configuration (1 rank per GPU), in order to isolate structural differences between both implementations.

In the gamma-only branch of `compute_becsum.f90` in subroutine `sum_bec` implemented in `PW/src/compute_becsum.f90`, the real-space workflow is:

- `invfft_orbital_gamma`
- `calbec_rs_gamma`

These routines are implemented in `PW/src/realus.f90` and involve:

- Distributed inverse FFTs,
- Real-space projector contractions,
- Device synchronization and memory updates.

3.9.1 Waiting-Time Behavior

Figures {3.3}, {3.5} and {3.4} show the evolution of t_{wait} for the real-space kernels.

In the figure {3.3} we can see that for `invfft_orbital_gamma` the rank 1 is the slowest, followed by rank 2. Ranks 0 and 3 are the fastest and their time is very similar. On the other hand, for `calbec_rs_gamma` rank 0 and 3 are the slowest and ranks 1 and 2 are the fast ones, their time is very similar. Here we can see that if there is opportunity to improve these algorithms to balance work and time. In figures {3.4} and {3.5} we can see in better detail what is going on inside each subroutine.

In contrast to the reciprocal-space case:

- Compute times is different.

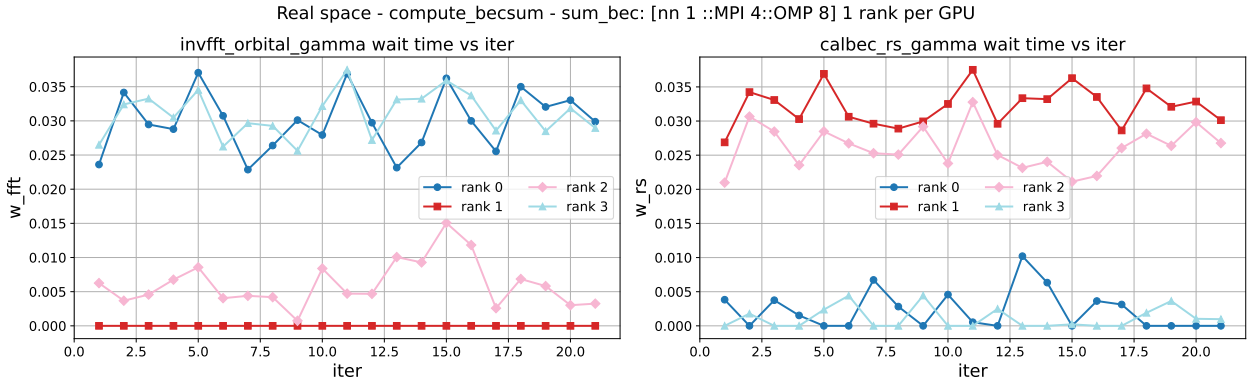


Figure 3.3: wait times for invfft_orbital_gamma and calbec_rs_gamma in compute_becsum/bec_sum for 1 ranks per GPU.

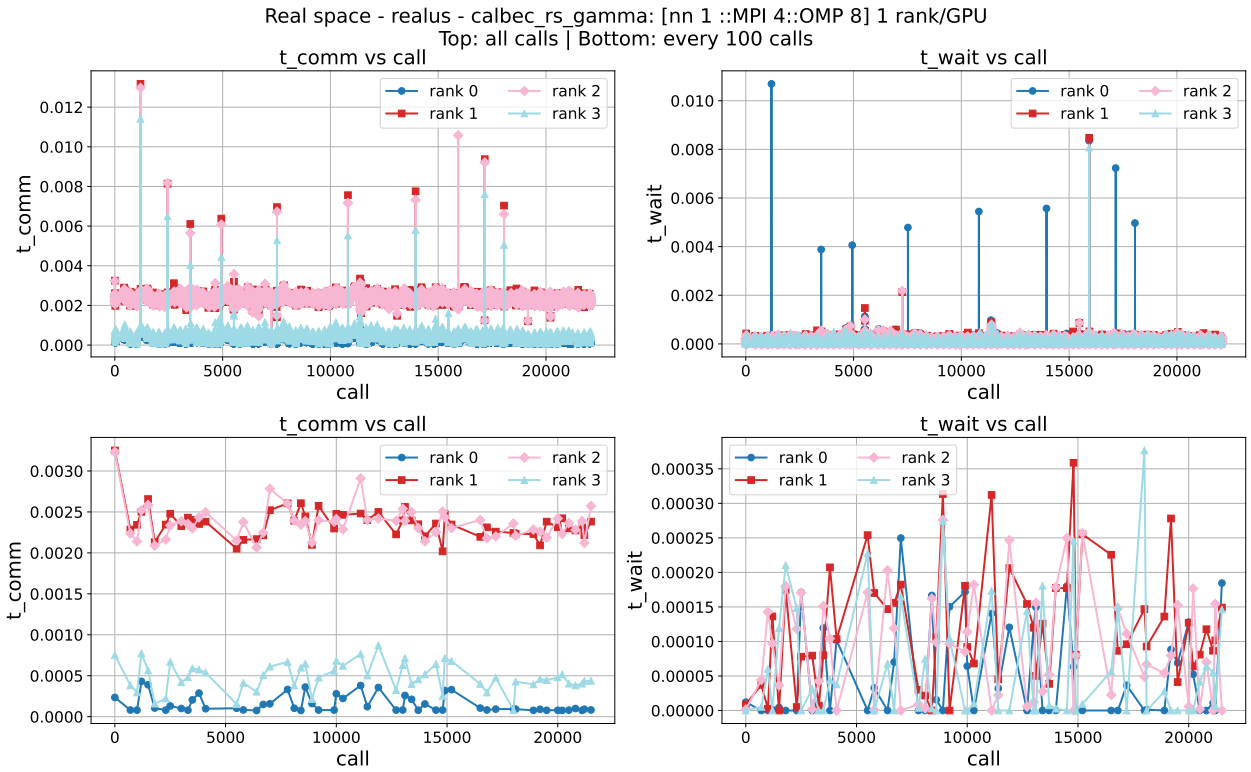


Figure 3.4: calbec_rs_gamma in realus.f90 for 1 ranks per GPU. The top part is for full data and the bottom part is for data every 100 calls.

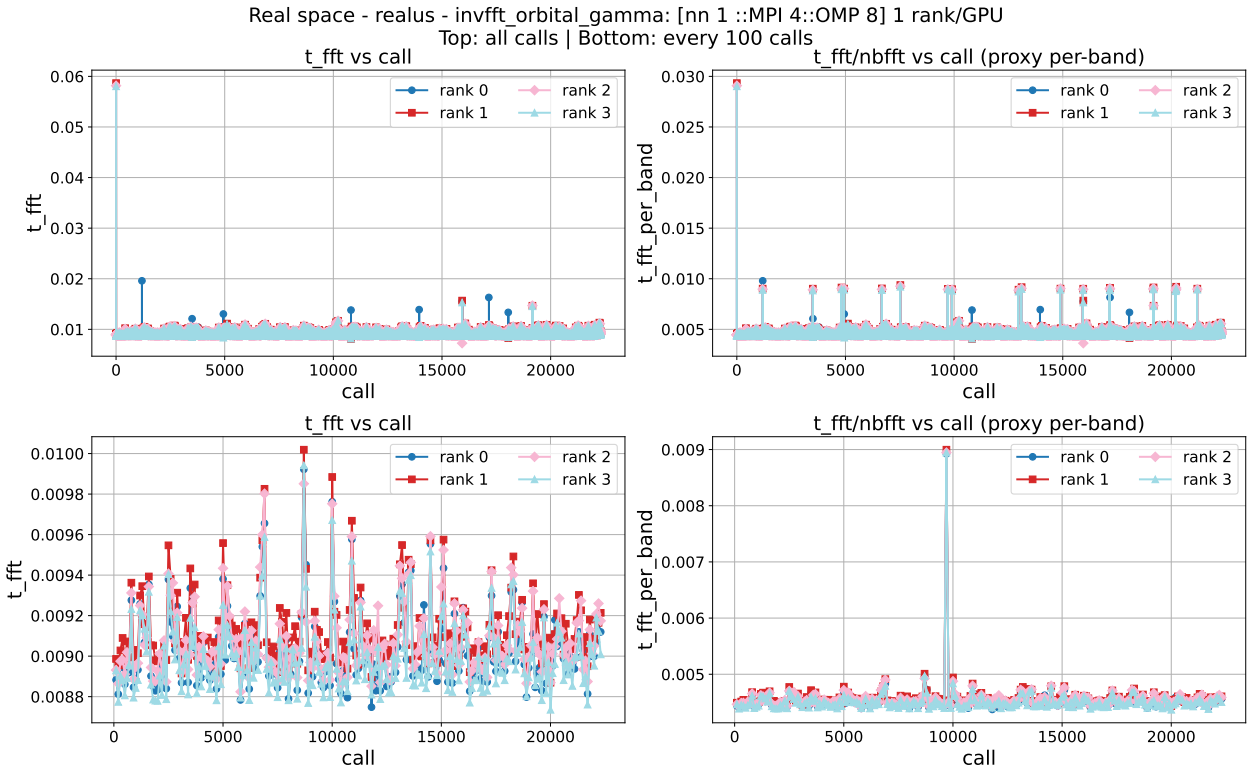


Figure 3.5: invfft_orbital_gamma in realus.f90 for 1 ranks per GPU. The top part is for full data and the bottom part is for data every 100 calls.

- t_{wait} exhibits recurrent spikes, but we can consider this as atypical data and in this way focus on the general state performance.
- The waiting time is unevenly distributed across ranks. An interesting fact is that the behavior is two to two, two slow and two fast with similar times.

Since the number of bands per group is identical, this imbalance cannot be explained purely by band partitioning.

3.9.2 Structural Origin of the Imbalance

The key structural differences are:

1. The inverse FFT requires global data redistribution from reciprocal sticks to real-space slabs.
2. Projector locality depends on grid decomposition.
3. Synchronization is introduced after FFT transposes and device updates.

Therefore, even when arithmetic workload is balanced, the effective execution time per rank may differ due to:

- Grid-slice distribution asymmetry,
- Communication latency in FFT transposes,
- Synchronization barriers across band groups.

The observed waiting time in real space is thus **structural** rather than purely band-driven.

3.9.3 Implications

The comparison between reciprocal and real space under identical hardware configuration (1 rank per GPU) reveals a clear distinction:

- Reciprocal space behaves predictably and remains balanced.
- Real space introduces synchronization-induced slack that is not proportional to band count.

This suggests that further optimization of real-space projectors requires algorithmic restructuring, for example:

- Reducing synchronization after FFT stages,
- Improving overlap between FFT communication and projector contractions,
- Reorganizing data layout to improve grid locality.

3.10 Conclusions and Summary

This chapter compared reciprocal-space and real-space projector pathways in Quantum ESPRESSO under a controlled configuration of 1 MPI rank per GPU, in order to isolate the intrinsic behavior of the kernels from artifacts caused by MPI oversubscription or intra-device contention.

The reciprocal-space projector path exhibits stable and well-balanced behavior: compute times are nearly identical across ranks, communication remains moderate, and waiting time is small and steady. This indicates that, in this regime, the reciprocal-space formulation does not introduce systematic synchronization slack, and its performance can be modeled primarily as a smooth function of the local band workload.

In contrast, the real-space projector path shows a clear and recurrent waiting-time asymmetry across ranks, often appearing as a paired pattern (two ranks consistently slower than the other two). Since the band distribution is identical across ranks in the analyzed configuration, this behavior cannot be attributed to band partitioning alone. Instead, it points to a structural origin linked to distributed FFT transposes, grid-slice locality of projector support, and synchronization semantics around collective communications and device updates.

These results lead to an important conclusion for the overall optimization strategy: while dynamic band redistribution is effective in mitigating solver-level imbalance, further improvements for large systems will require algorithmic work at the projector/FFT level, particularly for the real-space pathway.

It is important to note that such optimizations are beyond the scope of this thesis. The analysis presented here is based on detailed timings and performance characterization, and its purpose is to identify and isolate structural sources of imbalance in the real-space projector implementation. In this sense, the results provide clear evidence that additional load balancing and algorithmic improvements are required at the projector level.

Potential directions include improving overlap between FFT communication and contractions, revisiting batching and data layout, and reducing synchronization points in the real-space implementation.

Conclusions

This thesis investigated performance bottlenecks and load-imbalance mechanisms in band-parallel and projector-intensive workloads within Quantum ESPRESSO, with a primary focus on the linear-response solver `cgsolve_all` used by `phonon.x`, and a complementary analysis of reciprocal- and real-space projector paths in `pw.x`.

Dynamic band balancing in `cgsolve_all`. The first and central contribution of this work is the design and implementation of a dynamic band-balancing strategy that mitigates the intrinsic limitation of the static band distribution performed by `divide.f90`. While the default scheme assigns an almost identical number of bands to each band group, the true computational cost per band is heterogeneous and depends on both numerical and physical factors (e.g., conditioning of shifted operators in Sternheimer solves, near-degeneracies, and band-dependent convergence rates). As a consequence, the original solver exhibits inter-group waiting times at synchronization points, which become increasingly harmful as the number of band groups (BGs) grows.

The proposed method measures the band-group timing imbalance at runtime and redistributes bands adaptively between the slowest and fastest groups. The redistribution magnitude is controlled through a learning rate β and guardrails such as `QE_MAX_BAND_PROP`, ensuring stable and non-oscillatory convergence of the band allocation. Performance results on CPU and GPU platforms show that the approach improves strong-scaling behavior and reduces idle time, with gains that are system-dependent: heterogeneous systems (e.g., CuHO) achieve large reductions in CPU time (up to $\sim$ 20–27% in the tested configurations), while more homogeneous systems (e.g., SiO_2) show smaller but consistent improvements once multiple BGs are used. Importantly, because the optimization was implemented in the shared linear-response infrastructure (`LR_Modules`), the same benefits are expected to propagate to other QE executables relying on Sternheimer-based solvers beyond `PH.x`.

Projector-path analysis and structural imbalance. The second part of the thesis analyzed the computational structure of projector evaluation, contrasting reciprocal-space and real-space projector paths under a controlled configuration of 1 MPI rank per GPU. The reciprocal-space path shows predictable and well-balanced behavior: compute times are nearly identical across ranks, communication remains moderate, and the measured waiting time is small and stable. In contrast, the real-space path introduces a characteristic and recurrent waiting-time pattern that is not explained by band distribution alone.

The imbalance appears as a "two-fast / two-slow" grouping among ranks and is therefore indicative of structural effects tied to distributed FFT transposes, grid-slice locality, and synchronization around communication and device updates. While no direct modifications to the real-space implementation were performed within this work, the analysis provides clear evidence that the observed imbalance originates from the algorithmic structure of the projector evaluation.

This observation motivates future optimization efforts that go beyond band redistribution and target the algorithmic structure of real-space projectors.

Overall outcome. Taken together, the results support a general conclusion: improving scalability in large-scale DFPT and plane-wave workflows requires (i) dynamic workload redistribution at the band level for iterative solvers, and (ii) careful kernel-level analysis of projector and FFT pathways to expose structural sources of synchronization slack. The dynamic balancing strategy presented here is a practical and minimally invasive improvement that yields

immediate performance benefits, while the projector study highlights the next optimization frontier.

Several directions follow from this thesis:

- **Generalize dynamic balancing beyond pairwise transfers.** The current redistribution moves bands between the slowest and fastest BGs. Extending the policy to multi-way rebalancing (e.g., solving a small bounded optimization problem over all BGs) could accelerate convergence to a balanced configuration when many BGs are used.
- **Integrate more informative cost models.** The present method uses measured solve times as feedback. A hybrid model that combines timing with predictors (e.g., band-dependent iteration counts, residual histories, or energy proximity indicators) could reduce noise and improve responsiveness, especially in early iterations.
- **Reduce measurement overhead and improve reproducibility.** While runtime timing is essential, systematic strategies for sampling and smoothing (e.g., exponential moving averages) may improve stability across noisy runs and heterogeneous platforms.
- **Real-space projectors: algorithmic restructuring.** The projector analysis suggests that the bottleneck is structural: it emerges from the interaction between distributed FFT layouts, locality of real-space projectors, and synchronization semantics. Promising directions include: (i) improved overlap of FFT communication with projector contractions, (ii) revisiting the stick/slab redistribution and batching strategy, (iii) reordering and fusing kernels to reduce barriers and device sync points, and (iv) data-layout transformations that improve locality for the projector support regions on distributed grids.
- **Unify solver-level and kernel-level balancing.** Ultimately, the most effective approach may combine dynamic band redistribution with projector-aware scheduling, i.e., accounting for the fact that projector costs are not strictly proportional to the number of bands once real-space pathways and FFT collectives dominate.

In summary, the thesis provides both an immediately deployable optimization (dynamic band balancing) and a clear roadmap for deeper algorithmic work on real-space projector performance.

Bibliography

[1] R. M. Dreizler and E. K. U. Gross. Density Functional Theory: An Approach to the Quantum Many-Body Problem, volume 180 of Springer Series in Solid-State Sciences. Springer-Verlag, Berlin, Heidelberg, 1990. https://doi.org/10.1007%2F978-3-642-86105-5.

[2] Stefano Baroni, Stefano de Gironcoli, Andrea Dal Corso, and Paolo Giannozzi. Phonons and related crystal properties from density-functional perturbation theory. Rev. Mod. Phys., 73:515–562, Jul 2001. https://link.aps.org/doi/10.1103/RevModPhys.73.515.

[3] Quantum ESPRESSO. https://www.quantum-espresso.org/.

[4] M. Frigo and S.G. Johnson. The design and implementation of fftw3. Proceedings of the IEEE, 93(2):216–231, 2005. https://doi.org/10.1109/JPROC.2004.840301.

[5] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK Users' Guide. Society for Industrial and Applied Mathematics, third edition, 1999. https://doi.org/10.1137/1.9780898719604.

[6] Laura Susan Blackford, J. Choi, A. Cleary, A. Petitet, R. C. Whaley, J. Demmel, I. Dhillon, K. Stanley, J. Dongarra, S. Hammarling, G. Henry, and D. Walker. Scalapack: a portable linear algebra library for distributed memory computers - design issues and performance. In Proceedings of the 1996 ACM/IEEE Conference on Supercomputing, Supercomputing '96, page 5–es, USA, 1996. IEEE Computer Society. https://doi.org/10.1145/369028.369038,.

[7] A Marek, V Blum, R Johanni, V Havu, B Lang, T Auckenthaler, A Heinecke, H-J Bungartz, and H Lederer. The elpa library: scalable parallel eigenvalue solutions for electronic structure theory and computational science. Journal of Physics: Condensed Matter, 26(21):213201, may 2014. https://doi.org/10.1088/0953-8984/26/21/213201.

[8] General Public License. https://www.gnu.org/licenses/.

[9] John P. Perdew, J. A. Chevary, S. H. Vosko, Koblar A. Jackson, Mark R. Pederson, D. J. Singh, and Carlos Fiolhais. Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation. Phys. Rev. B, 46:6671–6687, Sep 1992. https://link.aps.org/doi/10.1103/PhysRevB.46.6671.

[10] Jianmin Tao, John P. Perdew, Viktor N. Staroverov, and Gustavo E. Scuseria. Climbing the density functional ladder: Nonempirical meta–generalized gradient approximation designed for molecules and solids. Phys. Rev. Lett., 91:146401, Sep 2003. https://doi.org/10.1103/PhysRevLett.91.146401.

[11] Richard Martin. Electronic Structure: Basic Theory and Practical Methods. 08 2020. https://www.amazon.com/Electronic-Structure-Theory-Practical-Methods/dp/0521534402.

[12] Warren E. Pickett. Pseudopotential methods in condensed matter applications. Computer Physics Reports, 9(3):115–197, 1989. https://www.sciencedirect.com/science/article/pii/0167797789900026.

[13] Attila Szabo and Neil S. Ostlund. Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory. Dover Publications, Mineola, NY, dover edition edition, 1996.

[14] Frank Jensen. Introduction to Computational Chemistry. Wiley, Chichester, 3rd edition, 2017.

[15] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. Phys. Rev., 136:B864–B871, Nov 1964. https://link.aps.org/doi/10.1103/PhysRev.136.B864.

[16] Pietro Delugas. Developer's Manual for PHonon (6.6). https://gitlab.com/gparedes137/q-e/-/blob/develop/PHonon/Doc/developer_man.pdf?ref_type=heads, 2021. [only partilly updated].

[17] INPUT_PH. https://www.quantum-espresso.org/Doc/INPUT_PH.html#id1.

[18] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. Journal of research of the National Bureau of Standards, 49:409–435, 1952. https://api.semanticscholar.org/CorpusID:2207234.

[19] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, USA, 1994. https://dl.acm.org/doi/book/10.5555/865018.

[20] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos. Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients. Rev. Mod. Phys., 64:1045–1097, Oct 1992. https://doi.org/10.1103/RevModPhys.64.1045.

[21] CINECA. https://www.hpc.cineca.it/.

[22] LEONARDO. https://www.hpc.cineca.it/systems/hardware/leonardo/.

[23] GALILEO100. https://www.hpc.cineca.it/systems/hardware/galileo100/.

[24] VTune Intel. https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler-download.html.

[25] Francesco Andreucci. Performance analysisvof GPU-to-GPU communications in distributed FFTs in Quantum ESPRESSO. Master thesis, Master in High Performance Computing, Trieste, Italy, 2024. Available in: https://iris.sissa.it/retrieve/101b3804-00c1-43ac-aeb3-14862ec42858/thesis_andreucci.pdf.