Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelorthesis

Optimization of the Quantum Espresso Density Functional Theory Code for parallel execution on the PHYSnet-Cluster

Optimierung des Quantum Espresso Dichte-Funktional Theorie Codes für die parallele Ausführung auf dem PHYSnet-Cluster

vorgelegt von

Tjark Sievers

Fakultät: Mathematik, Informatik und Naturwissenschaften
Fachbereich: Physik
Studiengang: Physik
Matrikelnummer: 7147558
Erstgutachter: Prof. Dr. Tim Wehling
Zweitgutachterin: Prof. Dr. Daria Gorelova

Kurzzusammenfassung

Diese Arbeit untersucht Quantum ESPRESSO, eine Sammlung von Programmen für Berechnungen von elektronischen Strukturen und Modellierung von Materialien in Bezug auf seine Skalierbarkeit über mehrere Prozessoren auf dem PHYSnet compute cluster. Die Methode ist eine Reihe an Benchmarks zum Testen von verschiedenen Compiler Kombinationen und den Parallelisierungsoptionen von Quantum ESPRESSO. Diese Benchmarks zeigen, dass die Nutzung von Compilern in Intel oneAPI die Skalierbarkeit signifikant verbessert, mit bis zu dreimal schnelleren Rechnungen auf einem Rechnerknoten. Außerdem ermöglicht die Nutzung der Parallelisierungsoptionen von Quantum ESPRESSO die Rechnungen weit über einen Rechnerknoten hinweg ermöglichen, wenn sie richtig genutzt werden. Ergebnisse aus den Benchmarks wurden außerdem genutzt, um effiziente Phononen Rechnungen von TaS_2 in einer Ladungsdichtewellephase duchzuführen, deren Ergebnisse möglicherweise eine Lücke um das Fermi-Niveau zu erklären, die 2019 in einem Scanning Tunneling Spectroscopy (STS) Experiment an diesem Material [1] gefunden wurde.

Abstract

This thesis examines Quantum ESPRESSO, a suite of computer code for electronic-structure calculations and materials modeling in terms of its scalability on multiple processors on the PHYSnet compute cluster. A series of benchmarks is carried out to test different combination of compilers as well as parallelization parameters offered by Quantum ESPRESSO itself. These benchmarks show that using a set of compilers and auxiliary code in Intel oneAPI significantly improves scaling, with up to three times faster calculations on a single compute node. Furthermore, the parallelization parameters offered by Quantum ESPRESSO let calculations scale beyond a single node when used right. Results from these benchmarks were then used to carry out efficient phonon calculations on TaS_2 in a charge density wave phase, the results of which could explain a gap feature near the Fermi level observed in a 2019 STS experiment on this material [1].

Contents

Glossary

BLAS (Basic Linear Algebra Subprograms) specification for routines that provide standard building blocks for performing basic vector and matrix operations. 14, 26

CPU time Time spent by a process on the supposed purpose. 16, 17

FFT (Fast Fourier Transform) Algorithm computing discrete Fourier transforms. 5, 14

Intel oneAPI Intel implementation of the oneAPI specification, providing among others MPI with C/Fortran compilers, implementations of the BLAS and LAPACK/ScaLAPACK APIs all optimized for Intel processors. i, 14, 26–34, 37, 40–42, 49

LAPACK (Linear Algebra Package) software package for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problem, using the BLAS routines. 14, 26

MPI (Message Passing Interface) communication protocol for programming parallel computers. 14, 26, 34

OpenBLAS Open source implementation of the BLAS and LAPACK APIs. 14, 26, 27

OpenMPI Open source MPI implementation. 14, 23–31, 49

ScaLAPACK (Scalable LAPACK) Implementation of a subset of LAPACK routines intended to use the advantages of running on parallel machines. 14, 16, 26, 27

wait time Time a process spent waiting while other processes run on the CPU. 16, 17, 23

wall time Real time elapsed between the start and end time of a process. 16, 17, 23, 31, 39

Acronyms

DFPT Density Functional Perturbation Theory. 5, 8

DFT Density Functional Theory. vii, 1–3, 5, 8, 19, 46

KS Kohn-Sham. 3, 4, 8, 9, 15, 16, 27, 33

PP Pseudopotentials. 5, 19, 21

STM Scanning Tunneling Microscope. 45, 46

STS Scanning Tunneling Spectroscopy. i, 46, 47, 49

TMDC Transition Metal Dichalcogenide. vii, 19

Motivation

For a realistic description of matter, methods derived from first principle (so called ab-initio methods) are needed. Phenomena explainable from ab initio methods span from thermodynamics properties of matter to superconductivity. The former deals with the description of quasi-particles emerging from the quantization of vibrational modes and the latter still lacks a theory explaining all kinds of known superconductivity.

One such ab-initio method is Density Functional Theory (DFT), the foundations of which were laid in 1964 by Hohenberg and Kohn [2], and in 1965 by Kohn and Sham [3]. Since 1990, methods within the density functional formalism have been very successful across a number of disciplines in physics, chemistry and biology, with over 160 000 publications on the topic between 1990 and 2015 [4]. The appeal of DFT methods lies in the fact that the complexity of calculations is reduced in such a way that objects such as the full wave function cannot be computed, total energies are very reliably produced, which in turn enables calculations of lattice dynamics, thermodynamical properties of matter or chemical reactions. These calculations are computationally cheap in comparison to methods working with full wave functions, so that simple systems can be simulated on a home computer today. In software suites such as Quantum ESPRESSO [5, 6] DFT methods are easily available today.

Going beyond simple calculations of a few atoms and towards current research questions makes parallel calculations over multiple nodes on compute cluster with hundreds or thousands of CPUs the only feasible possibility. An important step therefore is to guarantee that the process of scaling the work across multiple processors is done in an effective manner to utilize available computing resources as efficiently as possible.

Thus, the task of this thesis was two-fold: First, examining the way Quantum ESPRESSO calculations are best parallelized on the PHYSnet cluster and then using this knowledge to run calculations for a system of current interest and relate to recent experimental data of this system [1].

The examined system is TaS_2, a Transition Metal Dichalcogenide (TMDC). As bulk structures, TMDCs have been first described in 1923 by Dickinson and Pauling [7], in 1969 Wilson et al. characterized over 60 TMDCs. The more recently discovered monolayers of TMDCs [8] have brought a new focus on these materials, as they are among the candidates for materials enabling controllable electronic quantum phases [9]. TaS_2 in particular is notable as a superconducting material, both in the bulk and the monolayer case. Furthermore, both bulk and monolayer TaS_2 form a charge density wave (a periodic modulation of the electronic charge density of a solid) at low temperatures [1]. This particular phase of monolayer TaS_2 is an active area of research and will be examined in this thesis.

The structure of this thesis is as follows: first, all relevant theory needed to understand the calculations made with Quantum ESPRESSO will be outlined in chapter I. Following that, details regarding the computational work done, such as the concrete metrics evaluating

performance as well as a description of the parallelization parameters offered by QUANTUM ESPRESSO will be presented in chapter II. Chapter IV examines scalability of the `PWscf` module, which enables electronic structure calculations, the same is done in chapter V for the `PHonon` module, which is used for calculation of phonon and phonon related properties. The results from these chapters are then used to run an optimized phonon calculation on TaS_2 in the charge density wave phase. This optimized phonon calculation is then the foundation for a possible explanation of experimental data on TaS_2 in chapter VI.

Conventions

Throughout the text of this thesis scalars are written in italic s, vectors in bold italic $\boldsymbol{v}$ and matrices in bold $\mathbf{M}$ fonts. The summation/multiplication over nearest neighbour sites i and j is $\langle ij \rangle$ as a subscript to $\sum/\prod$. Furthermore, Hartree atomic units are used in general and only in selected instances it is deviated from this: $\hbar = m_\mathrm{e} = \mathrm{e} = {}^{4\pi}/_{\varepsilon_0} = 1$.

I Ab initio methods for materials modeling

The description of matter by theoretical methods starts from the Hamiltonian of an interacting system of electrons and nuclei (with electronic coordinates $\boldsymbol{r}_i$ and nucleic coordinates $\boldsymbol{R}_\alpha$)

$$\hat{H} = \hat{T}_\mathrm{e} + \hat{U}_\mathrm{e-e} + \hat{T}_\mathrm{n} + \hat{V}_\mathrm{n-e} + \hat{W}_\mathrm{n-n} \tag{I.1}$$

$$= -\sum_i \frac{1}{2}\nabla_i^2 + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} - \sum_\alpha \frac{1}{2M_\alpha}\nabla_\alpha^2 - \sum_{i,\alpha}\frac{Z_\alpha}{|\boldsymbol{r}_i - \boldsymbol{R}_\alpha|} + \frac{1}{2}\sum_{\alpha\neq\beta}\frac{Z_\alpha Z_\beta}{|\boldsymbol{R}_\alpha - \boldsymbol{R}_\beta|} \tag{I.2}$$

where

- $\hat{T}_\mathrm{e}\,, \hat{T}_\mathrm{n}$ are the kinetic energies of the electrons and nuclei respectively,
- $\hat{U}_\mathrm{e-e}$ is the Coulomb interaction between electrons,
- $\hat{V}_\mathrm{n-e}$ is the Coulomb interaction between electrons and nuclei,
- $\hat{W}_\mathrm{n-n}$ is the Coulomb interaction between nuclei.

This very general problem consisting of both the electronic and nucleic degrees of freedom can be simplified in a first step by employing the Born-Oppenheimer approximation [10]. The approximation assumes the nuclei to be fixed point charges which create a potential for the N interacting electrons, so that the electronic part can be solved independently using the nuclei positions as a parameter

$$\hat{H}_\mathrm{BO} = \hat{T}_\mathrm{e} + \hat{U}_\mathrm{e-e} + \hat{V}_\mathrm{n-e} + \hat{W}_\mathrm{n-n} \tag{I.3}$$

$$= -\sum_i \frac{1}{2}\nabla_i^2 + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\boldsymbol{r}_i - \boldsymbol{r}_j|} - \sum_i\sum_\alpha \frac{Z_\alpha}{|\boldsymbol{r}_i - \boldsymbol{R}_\alpha|} + \frac{1}{2}\sum_{\alpha\beta}\frac{Z_\alpha Z_\beta}{|\boldsymbol{R}_\alpha - \boldsymbol{R}_\beta|}\,. \tag{I.4}$$

The terms $\hat{V}_{n-e}$ and $\hat{W}_{n-n}$ here are just a function of the electronic coordinates $\boldsymbol{r}$ and a constant respectively. Hence they can then be combined into a potential $\hat{V}(\boldsymbol{r})$ for the interacting electrons and the Hamiltonian reads

$$\hat{H} = \hat{T} + \hat{U} + \hat{V}\,. \tag{I.5}$$

I.1 Density Functional Theory

Obtaining solutions to the Schrödinger equation with the Hamiltonian I.4 is analytically infeasible, as it produces a system of $3N$ coupled differential equations, with $N \sim N_\mathrm{Avogadro} \sim \mathcal{O}(10^{23})$. As such, the need for good approximations to obtain results for real world systems is high. One particularly successful approach is Density Functional Theory (DFT). In the

following section, the theoretical framework of DFT will be reviewed following the PhD thesis of Nicola Marzari [11], a more extensive discussion can be found in Richard Martins textbook on electronic structure [12].

I.1.1 Hohenberg-Kohn theorems

The start for DFT is the exact reformulation of the electronic structure problem by Hohenberg and Kohn. This reformulation uses the ground state density of the electronic system $n_0(r)$ as the basic variable. To achieve this, Hohenberg and Kohn formulated two theorems [2], which demonstrate that the ground state properties of an electronic system can be described using the ground state density (the proof of those theorems is omitted here, but can be found in the original publication [2] or the textbook by Martin [12, chapter 6.2]):

I The external potential is a unique functional of the ground state density.

II The ground state energy minimizes the energy functional,

$$E[n(\boldsymbol{r})] > E_0 \; \forall \; n(\boldsymbol{r}) \neq n_0(\boldsymbol{r})\,.$$

These theorems proof the existence and uniqueness of the energy functional $E[n(\boldsymbol{r})]$, but a concrete expression for it cannot be given. As the ground state wave function is a functional of the ground state density, a formal definition of the energy functional can be written as

$$\begin{aligned} E[n(\boldsymbol{r})] &= \langle\Psi|\hat{H}|\Psi\rangle \\ &= \langle\Psi|\hat{T}+\hat{U}+\hat{V}|\Psi\rangle \\ &= \langle\Psi|\hat{T}+\hat{U}|\Psi\rangle + \int \mathrm{d}\boldsymbol{r}'\,\Psi^*(\boldsymbol{r}')V(\boldsymbol{r}')\Psi(\boldsymbol{r}')\,. \end{aligned}$$

Here, $|\Psi\rangle$ is the full many-body state of the system at hand. Defining the universal functional $F[n(\boldsymbol{r})] = \langle\Psi|T+U|\Psi\rangle$, which is material independent and writing $n(\boldsymbol{r}') = \Psi^*(\boldsymbol{r}')\Psi(\boldsymbol{r}')$, the energy functional becomes

$$E[n(\boldsymbol{r})] = F[n(\boldsymbol{r})] + \int \mathrm{d}\boldsymbol{r}'\,V(\boldsymbol{r}')n(\boldsymbol{r}')\,. \tag{I.6}$$

This is just a formal definition, as all the formerly mentioned complication of the Hamiltonian I.5 now lies in the functional $F[n(\boldsymbol{r})]$. With a known or well approximated universal functional $F[n(\boldsymbol{r})]$, the Hohenberg-Kohn theorems provide a great simplification for finding the ground state properties of a solid state system, as the problem is now only a variational problem with three spatial coordinates instead of $3N$ coordinates when trying to solve the full Hamiltonian. This is due to the change to density as the basic variable instead of many-body wave-functions.

I.1.2 Kohn-Sham equations

Kohn and Sham proposed an approach for handling interacting many-body systems by introducing an auxiliary non-interacting system of electrons [3]

$$H_0 = \sum_i^{N_e} \frac{p_i^2}{2m} + v_{\mathrm{KS}}(\boldsymbol{r}_i)\,. \tag{I.7}$$

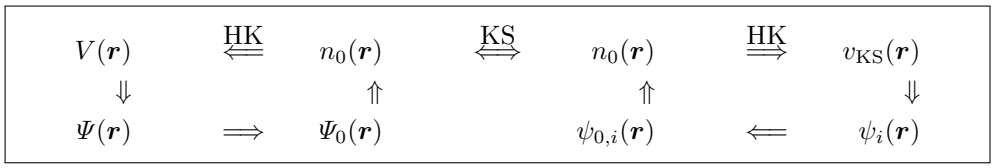


Figure I.1: *Representation of the KS ansatz. This scheme shows the connection between many-body properties (left) and the auxiliary KS system (right).* HK *here denotes the Hohenberg-Kohn theorems applied to the respective system. The schema is taken from [12, p. 137] and adapted for the notation in this thesis.*

With a correction potential v_{KS} such that the ground state charge density for the auxiliary and the interacting system are the same. This introduces a new set of orthonormal single-particle wave functions, the solutions to the non-interacting problem ψ_i. The density for this system is calculated as

$$n(\boldsymbol{r}) = \sum_{i=1}^{N_e/2} |\psi_i(\boldsymbol{r})|^2 \,. \tag{I.8}$$

The restriction of equality of the ground state densities for the interacting and the non-interacting system makes it possible to calculate all properties of the interacting system just by solving the non-interacting system (using the Hohenberg-Kohn theorems). This connection is visualized in a flowchart in fig. I.1. The important point is that in principle, all many-body properties are available through the Kohn-Sham (KS) scheme.

The kinetic energy of such a non-interacting problem can be easily calculated as sum over all electrons

$$T_{\mathrm{S}}[n(\boldsymbol{r})] = -\frac{1}{2}\sum_{i=1}^{N_e/2} \int \mathrm{d}\boldsymbol{r}\, \psi_i^*(\boldsymbol{r}) \Delta \psi_i(\boldsymbol{r}) \,. \tag{I.10}$$

With the help of this system and the classical electrostatic energy

$$E_{\mathrm{H}}[n(\boldsymbol{r})] = \frac{1}{2} \int\int \mathrm{d}\boldsymbol{r_1}\, \mathrm{d}\boldsymbol{r_2}\, \frac{n(\boldsymbol{r}_1) n(\boldsymbol{r}_2)}{|\boldsymbol{r}_1 - \boldsymbol{r}_2|} \,, \tag{I.11}$$

an ansatz for the total energy functional in eq. I.6 can be written as

$$E[n(\boldsymbol{r})] = T_{\mathrm{S}}[n(\boldsymbol{r})] + E_{\mathrm{H}}[n(\boldsymbol{r})] + E_{\mathrm{XC}}[n(\boldsymbol{r})] + \int \mathrm{d}\boldsymbol{r}'\, V(\boldsymbol{r}') n(\boldsymbol{r}') \,. \tag{I.12}$$

where now $E_{\mathrm{XC}}[n(\boldsymbol{r})]$ is a functional of the density accounting for all exchange and correlation effects not present in the non-interacting electron system. The success of DFT lies in the fact that E_{XC} contributes only a small part of the total energy and can be approximated in a useful manner. Even very simple approximations to E_{XC} such as the Local Density Approximation (LDA), which uses the exchange and correlation energy of a homogenous electron gas can give accurate results for very inhomogeneous systems [13]. Better results for systems with rapidly changing densities can be achieved with General-Gradient-Approximations (GGA) to E_{XC}, which also consider the gradient of the density. Perdew, Burke and Ernzerhof developed the first exchange-correlation-functional of this type , the PBE XC-functional [14].

Using that form of $E[n(\boldsymbol{r})]$, from the variational problem (with a Lagrange parameter introduced to ensure the orthonormality of the states ψ_i)

$$\delta\left(E[n(\boldsymbol{r})] - \sum_j \lambda_j \left[\int \mathrm{d}^3 r\, |\psi_j|^2 - 1\right]\right) = 0 \tag{I.13}$$

a set of single particle, Schrödinger-like equations can be derived:

$$\left(-\frac{1}{2}\Delta + \frac{1}{2}\int \mathrm{d}^3 r' \, \frac{n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} + \frac{\delta E_{\mathrm{XC}}}{\delta n(\boldsymbol{r})} + V\right)\psi_i(\boldsymbol{r}) = \lambda_i \psi_i(\boldsymbol{r}) \tag{I.14}$$

Schrödinger-like in this context means, that with the identification of the Hartree potential $V_{\mathrm{H}} = \frac{1}{2}\int \mathrm{d}^3 r' \, \frac{n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}$ and the exchange-correlation potential $V_{\mathrm{XC}} = \frac{\delta E_{\mathrm{XC}}}{\delta n(\boldsymbol{r})}$ the potential v_{KS} in eq. I.7 is

$$v_{\mathrm{KS}} = V_{\mathrm{H}} + V_{\mathrm{XC}} + V = \frac{1}{2}\int \mathrm{d}^3 r' \, \frac{n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|} + \frac{\delta E_{\mathrm{XC}}}{\delta n(\boldsymbol{r})} + V\,. \tag{I.15}$$

Eq. I.14 can be rewritten as the KS equations

$$\left(-\frac{1}{2}\Delta + v_{\mathrm{KS}}\right)\psi_i(\boldsymbol{r}) = \epsilon_i \psi_i(\boldsymbol{r})\,. \tag{I.16}$$

Importantly, the potential v_{KS} depends on the solutions $\psi_i(\boldsymbol{r})$, as V_{H} and V_{XC} include the density $n(\boldsymbol{r})$. The problem thus is a self-consistent problem, meaning the density used for calculating the potentials and the obtained solution only agree for the exact solution. Arriving at a solution consists then of iterating the process of obtaining a new set of potentials from the solution and solving the KS equations again.

This kind of iterative, self-consistent method lends itself to being implemented in a computational context, as every single step is mathematically simple and the complexity arises for instance from size of the matrices occurring or the number of steps needed, which makes calculation by hand tedious but is perfectly suited for execution on a computer.

I.1.3 Pseudopotentials and basis set

In order to represent the states and operators in eq. I.14, a basis set has to be chosen. Bloch's theorem states that in case of a periodic external potential, which makes the Hamiltonian commute with translation operators for translation by a lattice vector, the common eigenstates of these operators are:

$$\psi(\boldsymbol{r}) = \psi_{n\boldsymbol{k}}(\boldsymbol{r}) = e^{i\boldsymbol{k}\cdot\boldsymbol{r}} u_{n\boldsymbol{k}}(\boldsymbol{r})\,, \tag{I.17}$$

where $\boldsymbol{k}$ is the quasi-momentum, n is the band index and $u_{n\boldsymbol{k}}(\boldsymbol{r})$ has the periodicity of the unit cell. A natural choice to represent $u_{n\boldsymbol{k}}(\boldsymbol{r})$ is the discrete set of plane waves

$$u_{n\boldsymbol{k}}(\boldsymbol{r}) = \frac{1}{\sqrt{V}}\sum_{\boldsymbol{G}} e^{i\boldsymbol{G}\boldsymbol{r}}\,, \tag{I.18}$$

where $\boldsymbol{G}$ is a reciprocal lattice vector and V is the volume of the unit cell. From that the form $\psi_{n\boldsymbol{k}}$ follows:

$$\implies \psi_{n\boldsymbol{k}}(\boldsymbol{r}) = \frac{1}{\sqrt{V}} \sum_{\boldsymbol{G}} e^{i(\boldsymbol{k}+\boldsymbol{G})\boldsymbol{r}} \,. \tag{I.19}$$

With this choice of basis set, the kinetic energy is easily calculated:

$$\langle \psi_{n\boldsymbol{k}}(\boldsymbol{r})| - \nabla^2 |\psi_{n\boldsymbol{k}}(\boldsymbol{r})\rangle = \sum_{\boldsymbol{G}} c^2_{n\boldsymbol{k},\boldsymbol{G}} |\boldsymbol{k}+\boldsymbol{G}|^2 \,. \tag{I.20}$$

Another important consequence of this choice of basis set is that the electron density (eq. I.8) now becomes an integral over the Brillouin zone, which for numerical computation has to be approximated by a sum over a finite set of k points.

One problem of this choice of basis set lies in the fact that the lower energy core electrons are localized in the unit cell and as such need a significant amount of plane waves to be meaningfully described.

Importantly, the core electrons don't contribute directly to chemical bonds and physical properties of a solid, only by interaction with valence electrons. An approach to make the calculations more economical is then not to treat the core electrons explicitly, but instead to introduce an effective potential which includes the effects of both the core and the core electrons. These potentials are called Pseudopotentials (PP). They can be constructed from precise atomic calculations and significantly reduce the number of plane waves required while keeping the calculations accurate. One procedure to construct PPs which are transferable over a range of different environments was given by Haman et. al. [15]. The idea is to construct the potentials in such a way, that the energies from the real and pseudo wave functions (wave functions calculated with the PPs) agree, the real and pseudo wave functions agree beyond a certain radius r_C from the nucleus and the charge densities calculated from real and pseudo wave functions agree inside the radius r_C. PPs constructed in such a way are called norm-conserving PPs.

A *cutoff energy* E_{cutoff} can be used to further reduce computation time by allowing only expansion coefficient with

$$\frac{|\boldsymbol{k}+\boldsymbol{G}|^2}{2} \leq E_{\text{cutoff}} \,. \tag{I.21}$$

With Fast Fourier Transforms FFT, an efficient algorithm exists to transform from (discrete) real to (discrete) reciprocal space. Every expectation value can be calculated in the representation where the respective operators are diagonal: in reciprocal space for the kinetic energy and in real space for the potentials. Details about the implementation of this will be discussed in sec. II.2.2.

I.2 Density Functional Perturbation Theory

Within the framework of DFT a treatment of lattice vibrations can also be derived, as only knowledge of the ground-state density and its linear response to change in nucleic geometry is needed. Since this is the fundamental quantity in DFT, an extension in DFPT can be

developed for calculating properties of lattice vibrations. This theory will be outlined in this section, following the review article by Baroni et al. [16].

I.2.1 Sternheimer equation and Hellman-Feynman theorem

In a first step, two prerequisite equations are derived, namely the Sternheimer equation describing corrections to a wave function and the Hellman-Feynman theorem linking the derivative of the total energy of a system to the derivative of the Hamiltonian with regards to the same parameter.

The Sternheimer equation follows from perturbation theory. The idea is to treat a quantum system as an easily solvable system $\hat{H}_0$ experiencing a small perturbation $\hat{H}_1$. The Hamiltonian for the perturbed system is then

$$\hat{H} = \hat{H}_0 + \lambda \hat{H}_1 \tag{I.22}$$

with eigenstates und eigenvalues

$$\hat{H} \left|n\right\rangle = \epsilon_n \left|n\right\rangle \ . \tag{I.23}$$

These eigenstates and eigenvalues can now be expanded in terms of the parameter λ,

$$\left|n\right\rangle = \left|n^0\right\rangle + \lambda \left|n^1\right\rangle + \lambda^2 \left|n^2\right\rangle + \dots \ , \tag{I.24}$$

$$\epsilon_n = \epsilon_n^{(0)} + \lambda \epsilon_n^{(1)} + \lambda^2 \epsilon_n^{(2)} + \dots \ . \tag{I.25}$$

Inserting these expansion into eq. I.23 and sorting by order of λ^n gives then

$$0^{\text{th}} \text{ order:} \quad H_0 \left|n^0\right\rangle = \epsilon_n^{(0)} \left|n^0\right\rangle \tag{I.26}$$

$$1^{\text{st}} \text{ order:} \quad H_0 \left|n^1\right\rangle + H_1 \left|n^0\right\rangle = \epsilon_n^{(1)} \left|n^0\right\rangle + \epsilon_n^{(0)} \left|n^1\right\rangle \tag{I.27}$$

$$2^{\text{nd}} \text{ order:} \quad H_0 \left|n^2\right\rangle + H_1 \left|n^1\right\rangle = \epsilon_n^{(0)} \left|n^2\right\rangle + \epsilon_n^{(1)} \left|n^1\right\rangle + \epsilon_n^{(2)} \left|n^0\right\rangle \tag{I.28}$$

$$\vdots$$

Closing the scalar product in eq. I.27 with $\left\langle n^0\right|$ from the left leads to the first order energy correction via

$$\left\langle n^0\right| H_0 \left|n^1\right\rangle + \left\langle n^0\right| H_1 \left|n^0\right\rangle = \epsilon_n^{(1)} \left\langle n^0 | n^0\right\rangle + \epsilon_n^{(0)} \left\langle n^0 | n^1\right\rangle \tag{I.29}$$

$$\implies \epsilon_n^{(1)} = \left\langle n^0\right| H_1 \left|n^0\right\rangle \ . \tag{I.30}$$

The first order correction to the eigenstates, also known as the *Sternheimer equation* can be calculated by rearranging eq. I.27

$$(H_0 - \epsilon_n^{(0)}) \left|n^1\right\rangle = -(H_1 - \epsilon_n^{(1)}) \left|n^0\right\rangle \ . \tag{I.31}$$

The Hellman-Feynman theorem [17] links the derivative of the eigenvalue of a Hamiltonian $\hat{H}_\lambda$ depending on a continuos parameter λ with the derivative of the Hamiltonian with respect to that same parameter. Starting from the Schrödinger equation

$$\hat{H}_\lambda \left|\psi_\lambda\right\rangle = E_\lambda \left|\psi_\lambda\right\rangle \tag{I.32}$$

the derivative of E_λ with respect to λ can be calculated using the product rule

$$\frac{\partial E_\lambda}{\partial \lambda} = \frac{\partial}{\partial \lambda} \langle \psi_\lambda | \hat{H}_\lambda | \psi_\lambda \rangle \tag{I.33}$$

$$= \left\langle \frac{\partial \psi_\lambda}{\partial \lambda} \middle| \hat{H}_\lambda \middle| \psi_\lambda \right\rangle + \left\langle \psi_\lambda \middle| \hat{H}_\lambda \middle| \frac{\partial \psi_\lambda}{\partial \lambda} \right\rangle + \left\langle \psi_\lambda \middle| \frac{\partial \hat{H}_\lambda}{\partial \lambda} \middle| \psi_\lambda \right\rangle . \tag{I.34}$$

$\hat{H}_\lambda$ can now act on the states $|\psi_\lambda\rangle$ to the right or to the left in the first two terms

$$= E_\lambda \left\langle \frac{\partial \psi_\lambda}{\partial \lambda} \middle| \psi_\lambda \right\rangle + E_\lambda \left\langle \psi_\lambda \middle| \frac{\partial \psi_\lambda}{\partial \lambda} \right\rangle + \left\langle \psi_\lambda \middle| \frac{\partial \hat{H}_\lambda}{\partial \lambda} \middle| \psi_\lambda \right\rangle \tag{I.35}$$

$$= E_\lambda \frac{\partial}{\partial \lambda} \langle \psi_\lambda | \psi_\lambda \rangle + \left\langle \psi_\lambda \middle| \frac{\partial \hat{H}_\lambda}{\partial \lambda} \middle| \psi_\lambda \right\rangle \tag{I.36}$$

$$= E_\lambda \frac{\partial}{\partial \lambda} 1 + \left\langle \psi_\lambda \middle| \frac{\partial \hat{H}_\lambda}{\partial \lambda} \middle| \psi_\lambda \right\rangle . \tag{I.37}$$

With that the *Hellman-Feynman theorem* follows:

$$\frac{\partial E_\lambda}{\partial \lambda} = \left\langle \psi_\lambda \middle| \frac{\partial \hat{H}_\lambda}{\partial \lambda} \middle| \psi_\lambda \right\rangle \tag{I.38}$$

I.2.2 Lattice vibrations from electronic structure

As discussed in the beginning of this chapter, nucleic and electronic degrees of freedom can be decoupled in the Born-Oppenheimer approximation. The connection between lattice dynamics and the electronic structure in the Born-Oppenheimer approximation was first pointed out by De Cicco and Johnson [18] and Pick, Cohen, and Martin [19]. The lattice dynamics are determined by the Schrödinger equation involving the previously neglected kinetic energy of the nuclei and $E(\boldsymbol{R})$, the ground-state energy of the electronic Hamiltonian I.4, which depends parametrically on the set of all nucleic coordinates $\boldsymbol{R}$:

$$\left(-\sum_\alpha \frac{1}{2M_\alpha} \nabla_\alpha^2 + E(\boldsymbol{R}) \right) \Phi(\boldsymbol{R}) = \mathcal{E} \Phi(\boldsymbol{R}) \tag{I.39}$$

This equation gives eigenstates $\Phi(\boldsymbol{R})$ and energies $\mathcal{E}$ for the nuclei. The system is then in equilibrium, when the forces acting on the nuclei vanish

$$\boldsymbol{F}_\alpha = -\frac{\partial E(\boldsymbol{R})}{\partial \boldsymbol{R}_\alpha} = 0 . \tag{I.40}$$

The vibrational frequencies ω are determined by the Hessian of $E(\boldsymbol{R})$, usually called the matrix of interatomic force constants:

$$\det \left| \frac{1}{\sqrt{M_\alpha M_\beta}} \frac{\partial^2 E(\boldsymbol{R})}{\partial \boldsymbol{R}_\alpha \partial \boldsymbol{R}_\beta} - \omega^2 \right| = 0 \tag{I.41}$$

Thus, the calculation of the vibrational properties as well as the equilibrium geometry depend on the first and second derivatives of the energies $E(\boldsymbol{R})$. These derivatives can be calculated using the Hellman-Feynman theorem I.38, where the parameter λ is the nucleic coordinate $\boldsymbol{R}_\alpha$. Keeping in mind that in the Born-Oppenheimer Hamiltonian I.4 only the Coulomb interaction between the electrons and the nuclei and the Coulomb interaction between the nuclei have a dependence on the nucleic coordinates $\boldsymbol{R}$, the force acting on nucleus α is then

$$\boldsymbol{F}_\alpha = -\frac{\partial E(\boldsymbol{R})}{\partial \boldsymbol{R}_\alpha} = -\left\langle \Psi_{\boldsymbol{R}}(\boldsymbol{r}) \middle| \frac{\partial \hat{H}_{\mathrm{BO}}(\boldsymbol{R})}{\partial \boldsymbol{R}_\alpha} \middle| \Psi_{\boldsymbol{R}}(\boldsymbol{r}) \right\rangle \tag{I.42}$$

$$= -\left\langle \Psi_{\boldsymbol{R}}(\boldsymbol{r}) \middle| \frac{\partial \hat{V}_{\mathrm{n-e}}}{\partial \boldsymbol{R}_\alpha} + \frac{\partial \hat{W}_{\mathrm{n-n}}}{\partial \boldsymbol{R}_\alpha} \middle| \Psi_{\boldsymbol{R}}(\boldsymbol{r}) \right\rangle \tag{I.43}$$

$$= -\int \mathrm{d}\boldsymbol{r}\, n_{\boldsymbol{R}}(\boldsymbol{r}) \frac{\partial V_{\boldsymbol{R}}(\boldsymbol{r})}{\partial \boldsymbol{R}_\alpha} - \frac{\partial W_{\mathrm{n-n}}}{\partial \boldsymbol{R}_\alpha}\,, \tag{I.44}$$

where $\Psi_{\boldsymbol{R}}(\boldsymbol{r})$ is the ground state wave function of $\hat{H}_{\mathrm{BO}}(\boldsymbol{R})$ and $n_{\boldsymbol{R}}(\boldsymbol{r})$ is the ground state electronic density corresponding to a nucleic configuration $\boldsymbol{R}$.

The second derivative of $E(\boldsymbol{R})$ is then calculated using the product rule

$$\frac{\partial^2 E(\boldsymbol{R})}{\partial \boldsymbol{R}_\alpha \partial \boldsymbol{R}_\beta} = -\frac{\partial \boldsymbol{F}_\alpha}{\partial \boldsymbol{R}_\beta} = \int \mathrm{d}\boldsymbol{r}\, \frac{\partial n_{\boldsymbol{R}}(\boldsymbol{r})}{\partial \boldsymbol{R}_\beta} \frac{\partial V_{\boldsymbol{R}}(\boldsymbol{r})}{\partial \boldsymbol{R}_\alpha} + \int \mathrm{d}\boldsymbol{r}\, n_{\boldsymbol{R}}(\boldsymbol{r}) \frac{\partial^2 V_{\boldsymbol{R}}(\boldsymbol{r})}{\partial \boldsymbol{R}_\alpha \partial \boldsymbol{R}_\beta} + \frac{\partial^2 W_{\mathrm{n-n}}}{\partial \boldsymbol{R}_\alpha \partial \boldsymbol{R}_\beta}\,. \tag{I.45}$$

The lattice dynamics are thus determined by the electronic density $n(\boldsymbol{r})$ and its linear response to a change in the nuclear geometry $\frac{\partial n_{\boldsymbol{R}}(\boldsymbol{r})}{\partial \boldsymbol{R}_\beta}$.

I.2.3 Density Functional Perturbation Theory

The density response $\frac{\partial n_{\boldsymbol{R}}(\boldsymbol{r})}{\partial \boldsymbol{R}_\beta}$ can be calculated within a KS DFT formulation. This approach combining perturbation theory, linear response theory and DFT is called Density Functional Perturbation Theory (DFPT), developed by Baroni et al. [20] and Gonze [21].

In a first step, the electronic density I.8 will be linearized

$$\Delta n(\boldsymbol{r}) = 4\,\mathrm{Re}\left\{ \sum_i^{N_\mathrm{e}/2} \psi_i^*(\boldsymbol{r}) \Delta\psi_i(\boldsymbol{r}) \right\} \tag{I.46}$$

with the finite-difference operator regarding the parameter $\boldsymbol{R}$ (the superscript has been omitted in eq. I.46)

$$\Delta^{\boldsymbol{R}} F = \sum_\alpha \frac{\partial F_{\boldsymbol{R}}}{\partial \boldsymbol{R}_\alpha} \Delta \boldsymbol{R}_\alpha\,. \tag{I.47}$$

The variation of the KS orbitals can be obtained with the Sternheimer equation I.31

$$(H_0 - \epsilon_i)\,|\Delta\psi_i\rangle = -(\Delta v_{\mathrm{KS}} - \Delta\epsilon_i)\,|\psi_i\rangle\,, \tag{I.48}$$

where H_0 is the unperturbed KS Hamiltonian I.7, Δv_{KS} is the first-order correction to the potential v_{KS}

$$\Delta v_{\mathrm{KS}} = \frac{1}{2}\int \mathrm{d}^3 r' \frac{\Delta n(\boldsymbol{r}')}{|\boldsymbol{r} - \boldsymbol{r}'|} + \left.\frac{\partial v_{\mathrm{XC}}}{\partial n}\right|_{n=n(\boldsymbol{r})} + \Delta V \tag{I.49}$$

and $\Delta\epsilon_i = \langle\psi_i|\Delta v_{\mathrm{KS}}|\psi_i\rangle$ is the first-order correction to the KS eigenvalue ϵ_i.

As the right hand site in eq. I.48 depends again on the perturbed density $\Delta n(\boldsymbol{r})$ (and as such on $|\Delta\psi_i\rangle$), eq. I.46, I.48 and I.49 are again self-consistent equations. Wherefore, they can be solved in an iterative manner.

II Computational Details

II.1 Parallel computing and scalability

The following section will give an overview of the technical aspects of running computer code, such as QUANTUM ESPRESSO, on massively parallel computing environments. The information presented in this section follows closely the textbook on high-performance computing by Hager and Wellein [22].

In scientific computing, one can identify two distinct reasons for distributing workload to multiple processors:

- The execution time on a single core is not sufficient. The definition of sufficient is dependent on the specific task and can range from "over lunch" to "multiple weeks".
- The memory requirements exceed the capabilities of a single core.

Parallelization of a task across multiple processors can be distinguished into two ways:

Single Program Multiple Data (SPMD) Every processor runs the same program, with data distributed among processors.

Multiple Program Multiple Data (MPMD) Every processor runs a different function, for example in a pipelining process where multiple consequent operations on the input data need to be done and data comes in chunks, so every step of the pipeline can run independent of the others.

The typical case in physics is SPMD. For instance, many calculations require diagonalization of matrices, which can be iteratively done with algorithms requiring only knowledge of the data of the nearest neighbors for every matrix element in every step. This enables parallelization as the whole matrix can be distributed and communication is only required at the bordering regions for exchange of data in every iteration step. In the case where the iteration step in one region is faster than in another, waiting times are introduced, as the next iteration step can only be done after data exchange with the neighboring regions.

In order to judge how well a task can be parallelized, a scalability metric is employed, for example:

- How fast can a problem be solved with N processors instead of one?
- What kind of bigger problem (finer resolution, more particles, etc.) can be solved with N processors?
- How efficiently are the resources utilized?

In this thesis, the main concern is speeding up the execution of extensively expensive calculations with a fixed problem size, so the first metric will be used to judge the quality of parallelization.

This metric is called speedup and is defined as

$$S \coloneqq \frac{T_1}{T_N}\,, \tag{II.1}$$

where T_1 is the execution time on a single processor and T_N is the execution time on N processors. In the ideal case, where all the work can be perfectly distributed among the processors and all processors need the same time for their respective workload, the execution time on N processors would be T_1/N, so inserting this into eq. II.1 gives a speedup of

$$S = \frac{T_1}{\frac{T_1}{N}} = N\,. \tag{II.2}$$

In reality, there are many factors either limiting or in some cases supporting parallel code scalability. Limiting factors include:

Algorithmic limitations When parts of a calculation are mutually dependent on each other, the calculation cannot be fully parallelized.

Bottlenecks In any computer system exist resources which are shared between processor cores with limitations on parallel access. This serializes the execution by requiring cores to wait for others to complete the task which uses the shared resources in question.

Startup Overhead Introducing parallelization into a program necessarily introduces an overhead, e.g. for distributing data across all the processors.

Communication Often solving a problem requires communication between different cores (e.g. exchange of interim results after a step of the calculation). Communication can be implemented very effectively, but can still introduce a big prize in computation time.

On the other hand, *better caching* can lead to better scaling than $S = N$: as optimal performance per core is achieved when all the data can be kept in cache, reducing the data size per processor by distributing data among more processors can lead to each individual processor being faster than in the single core case.

A simple ansatz for modeling speedup with these limitations in mind was first derived by Gene Amdahl [23]. Assuming the work that needs to be done is split into a part which cannot be parallelized s and a part which can be parallelized ideally p, serial time can be normalized to 1:

$$T_1 = s + p = 1 \tag{II.3}$$

The time for solving the problem on N processors is then

$$T_N = s + \frac{p}{N}\,. \tag{II.4}$$

The speedup is now

$$S = \frac{T_s}{T_p} = \frac{1}{s + \frac{p}{N}} = \frac{1}{s + \frac{1-s}{N}}\,. \tag{II.5}$$

This equation is called *Amdahl's law*. It shows that even for $N \to \infty$, the speedup has an upper bound of $\frac{1}{s}$. Furthermore, the value of s determines the range of processors where the speedup is close to the ideal case, as shown in fig. II.1. It shows that for a bigger value of s, not only leads to the speedup saturating at a smaller constant, but also differing significantly

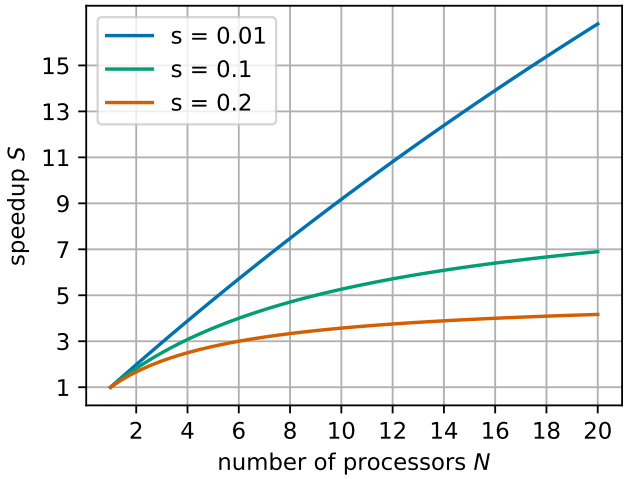


Figure II.1: *Speedup modeled by Amdahl's law for different portions of strictly serial workload*

from the ideal case even for a small number of processors. For $s = 0.01$, a speedup of around 16 for 20 processors used can be deemed acceptable in terms of how efficient the computing resources are used, whereas even using more than 6 processors in the case of $s = 0.2$ is not.

The weakness of Amdahl's law lies in the simplicity of it, as the different factors limiting parallelization are generally not independent of N. Communication overhead would need to be accounted for with some kind of function $c(N)$ with the form depending on many factors like speed and bandwidth of the communication hardware or the way the data is distributed.

With just Amdahl's law, decomposition into the several factors limiting parallelization is not possible, so an assessment of how calculations can be improved in detail is also not possible. Regardless, Amdahl's law explains in simple ways how speedup can differ from the ideal case and can also be reasonable accurate when the costs for communication don't depend strongly on N.

II.2 Quantum ESPRESSO

Quantum ESPRESSO (opEn-Source Package for Research in Electronic Structure, Simulation, and Optimization) [5, 6] is a collection of packages implementing (among others) the techniques described in sec. I.1 and I.2 to calculate electronic structure properties (module `PWscf`) as well as phonon frequencies and eigenmodes (module `PHonon`).

II.2.1 Compilation of Quantum ESPRESSO

As motivated above, the main goal of this thesis is an in-depth analysis of the QE software with respect to performance in terms of used computation resources. The choice and availability of different compilers will significantly influence this. Therefore, a short overview is presented

here. The information in this section is taken from the QUANTUM ESPRESSO 7.0 user guide [24].

The QUANTUM ESPRESSO distribution is packaged with everything needed for simple, non-parallel execution, the only additional software needed are a minimal Unix environment (a shell like `bash` or `sh` as well as the utilities `make` `awk` and `sed`) and a Fortran compiler compliant with the F2008 standard. For parallel execution, also MPI libraries and an MPI aware compiler need to be provided.

QUANTUM ESPRESSO needs three external mathematical libraries, BLAS and LAPACK for linear-algebra as well as an implementation of FFT for Fourier transforms. In order to make the installation as easy as possible, QUANTUM ESPRESSO comes with a publicly available reference implementation of the BLAS routines, the publicly available LAPACK package and an older version of FFTW (Fastest Fourier Transform in the West, an open source implementation of FFT). Even though these libraries are already optimized in terms of the algorithms implemented, usage of libraries implementing the same routines which can use more specific CPU optimizations might improve performance, e.g. libraries included in Intel oneAPI, which are optimized for use on Intel CPUs.

On the PHYSnet cluster, a variety of software packages are available as modules. The benchmarks in this thesis were made using the following module combinations:

- `openmpi/4.1.1.gcc10.2-infiniband`: OpenMPI 4.1.0 (implies usage of QUANTUM ESPRESSO provided BLAS/LAPACK)
- `openmpi/4.1.1.gcc10.2-infiniband openblas/0.3.20`: OpenMPI 4.1.0 and OpenBLAS 0.3.20
- `scalapack/2.2.0`: OpenMPI 4.1.0, OpenBLAS 0.3.20 and ScaLAPACK 2.2.0
- `intel/oneAPI-2021.4`: Intel oneAPI 2021.4

QUANTUM ESPRESSO offers a configuration script to automatically find all required libraries. As the default options of the `configure` script work well in the use case of this thesis, all compilations were made using the minimal commands

```
module load <module names>
./configure --with-scalapack=no|yes|intel
```

with the scalapack options yes (when using `scalapack/2.2.0`), intel (when using `intel/oneAPI-2021.4`) and no otherwise.

The output of the configuration script gives information about the detected libraries. In the following output, the Intel Intel oneAPI package was loaded, so BLAS and ScaLAPACK libraries from that package will be used, whereas the included FFT library will be used:

```
The following libraries have been found:
BLAS_LIBS=  -lmkl_intel_lp64  -lmkl_sequential -lmkl_core
LAPACK_LIBS=
SCALAPACK_LIBS=-lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64
FFT_LIBS=
```

II.2.2 Parallelization capabilities implemented in Quantum ESPRESSO

QUANTUM ESPRESSO is intended to be used in parallel environments and as such offers possibilities to manage how the work is parallelized. This section introduces the parallelization capabilities of the `PWscf` and `PHonon` modules and explores how they potentially affect the scaling behavior of QUANTUM ESPRESSO. The information in this section stems from the user guides for the two modules [25, 26]

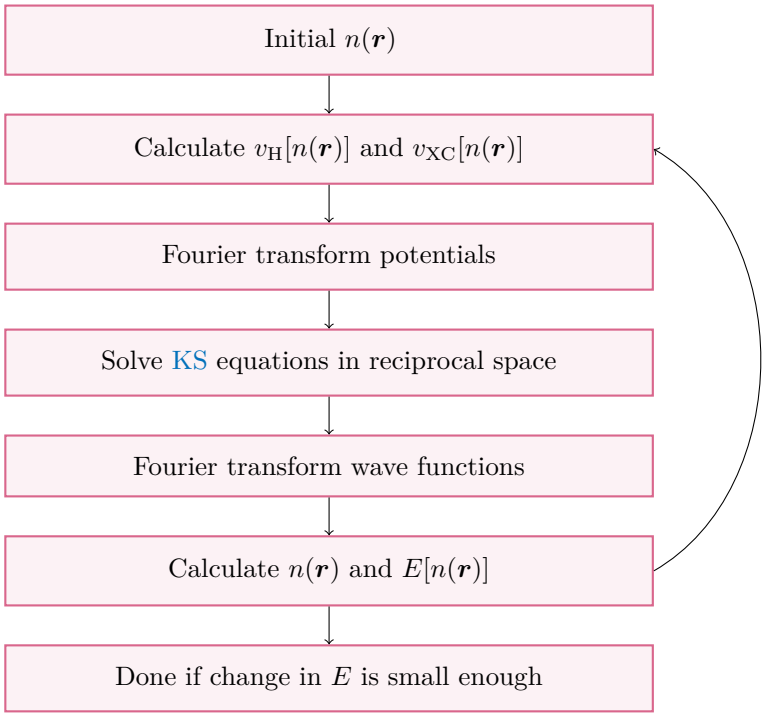


Figure II.2: *Flowchart of an iterative algorithm to solve the KS equations with the use of Fourier transform. As the density $n(\boldsymbol{r})$ determines again the potentials going into the KS equations, steps 2-5 are run until self-consistency is reached.*

Fig. II.2 shows a possible approach to solving the KS equations. The algorithm is taken from the textbook by Martin [12], with the Fourier transform steps added to establish in which representation every calculation step is run.

A few possibilities for parallelization of calculations can be derived from that. First of all, the real and reciprocal space are discretized to allow for numerical treatment, these grids can be distributed among processors, meaning the wave functions in the plane-wave basis set as well as charges and densities. This distribution of data mainly works around memory constraints, as using more processors lowers the memory requirement for every single processor. Going further, QUANTUM ESPRESSO automatically parallelizes all linear-algebra operations on this real space/reciprocal grid. The price to pay for this parallelization is the need for communication between processors: as an example, Fourier transforms always need to collect and distribute

contributions from and to the whole reciprocal/real grid in order to transform between them. This kind of parallelization is called *PW (plane-wave)* or *R&G (real & reciprocal)* parallelization.

As discussed in sec. I.1.3, the density in the plane-wave basis set is a sum over different k points, where the calculation for these are independent of each other until calculating the density $n(\boldsymbol{r})$. In Quantum ESPRESSO this is implemented such that a separation of the total number of processors into smaller pools, each doing the calculations for a set of k points is possible. This is called *k-point parallelization*. The CLI parameter `-nk <number of pools>` determines how many pools the total number of processor N is split into. Hence, the resulting number of processors in one pool is N/N_k. Within one k-point processor pool, the PW parallelization with its heavy communication is automatically applied.

On a level of parallelization independent of that, Quantum ESPRESSO can use ScaLAPACK to parallelize (among other things) the iterative orthonormalization of KS states. This parallelization level is called *linear-algebra parallelization* and is controlled by the CLI parameter `-nd <number of processors in linear-algebra group>`. Importantly, this parameter sets the size for the linear-algebra group in every k-point processor pool, so the number of processors in the linear-algebra group has to be smaller than the number of processors in one pool. Furthermore, the arrays on which the calculations are performed on are distributed in a 2D grid among processors. This means that the number of processors in the linear-algebra group has to be a square number.

In the case of the `PHonon` module, the representation of states in a plane-wave basis set stays the same, so all three parallelization schemes mentioned for the `PWscf` module can also be employed. Furthermore, as calculations for two phonon wave vectors $\boldsymbol{q}, \boldsymbol{q}'$ are not coupled (as different wave vectors lead to different perturbations and as such independent self-consistent equations), they can be split up into images. The concept of image parallelization in Quantum ESPRESSO is actually more general than just for phonon calculations, as other kinds of independent iterative calculations can also be run with image parallelization. The parameter controlling image parallelization is `ni <number of images>`. Following this, the number of processors in one k-point pool is then given by $N/N_i/N_k$, if image and k-point parallelization is applied, where N_i denotes the number of images.

II.2.3 Evaluating the scalability of Quantum ESPRESSO calculations

In the Quantum ESPRESSO output, a time report is printed at the end. This time report includes CPU time and wall time. Three different metrics of scalability can be calculated from this:

- runtime: absolute runtime (wall time) of the compute job
- speedup: runtime on N processors divided by runtime on a single core
- wait time: percentage of wall time not used by Quantum ESPRESSO process, so writing to disk, waiting for IO devices or other processes, etc. (calculated as (wall time - CPU time) / wall time)

For analysis mainly speedup will be used to evaluate the scalability of Quantum ESPRESSO calculation. It makes comparing the scaling of calculations with different absolute runtimes easy: as discussed in sec. II.1, optimal scaling is achieved when the speedup has a slope of one, independent of the runtime.

Regardless, the other two parameters should also always be considered. In the end, absolute runtime is the most important factor and should govern the decision of how much computational resources should be used for solving a particular problem. For instance, a problem with a single core runtime of 600 s, a speedup of 100 would mean a runtime of 6 s, whereas a speedup of 200 would mean a runtime of 3 s. Even with optimal scaling, the 100 processors needed for the speedup of 200 could be considered wasted for just 3 s of saved time. On the other hand, for a problem with a single core runtime of 2400 h, the difference between a speedup of 100 (runtime 24 h) and 200 (runtime 12 h) is the difference between waiting a whole day for the calculation and being able to let the job run overnight and continue in the morning, so using 100 more processors for that can be considered a good use of resources.

As for the wait time, this metric can be used to separate the different factors of poor parallelization discussed in II.1. Startup overhead is easy to identify, as this should be a small, near constant percentage of the absolute runtime. This of course can vary depending on how complex data distribution is, but there should at least not be a strong dependence on the number of processors, as only a small amount of communication is needed. Communication and bottlenecks on the other hand both introduce wait time which depends on the number of processors. Differentiating between them relies on knowledge of the specific hardware of the system running the calculations. This means how many cores are on a single chip, motherboard or node, which resources are shared between how many cores etc..

For this interpretation to be meaningful, the CPU and wall times reported by Quantum ESPRESSO have to be accurate. As an example for how errors could be overseen, when executing programs on multiple processors in parallel, CPU time is measured per processors. This means some kind of information truncation is done when a single number (such as in Quantum ESPRESSO) is reported. Whether this is taking the average over all processors, just reporting the time for a single processor or any other kind of truncation is unclear.

However, the notion of using the difference between wall time and CPU time for evaluating the quality of parallelization is supported by the user guide for one of the Quantum ESPRESSO modules [25] (sec. 4.5), therefore it will also be used as a qualitative measure of good parallelization in this thesis.

II.3 Hardware configuration of the PHYSnet cluster

All calculations were run on a reserved subset of the `infinix` queue on the PHYSnet compute cluster with 20 nodes. As of time of writing, the nodes in this queue are equipped with two Intel Xeon E5-2680 CPUs, as such providing 20 cores per node, 10 per chip and 200 processors in the whole queue. The nodes are connected with an Infiniband FDR 4x network.

III Examined systems

III.1 Silicon

Silicon is the fundamental material in modern transistors and as such one of the pillars of the digital revolution. Analogue to the Stone, Bronze or Iron Age, the current age of civilization can thus be called the Silicon Age [27]. Consequently, silicon is a well studied material from an experimental and theoretical standpoint. Combining this with the fact that DFT calculations on silicon are not particularly expensive makes it an ideal system for an introduction to DFT calculations as well as a good benchmarking system. Consequently, all benchmarks in this thesis were run on silicon first and with the information gained from that, benchmarks on a more expensive system were run.

The calculations in chapter IV and V were made with a plane-wave cutoff of 70 Ry and on a $40 \times 40 \times 40/6 \times 6 \times 6$ k-point grid respectively. All calculations use a PBE XC-functional with a norm-conserving PP generated using Vanderbilt's method [28].

III.2 TaS_2

Tantalum Disulfide (TaS_2) belongs to the class of Transition Metal Dichalcogenide (TMDC)s. The most common stoichiometry of compounds in this class is MX_2, where M is a transition-metal and X is a chalcogen atom. TMDCs occur with different atomic coordinations. Fig. III.1 shows the structure of trigonal-prismatic TaS_2 (2H-TaS_2), which consists of a hexagonal transition-metal lattice between two hexagonal chalcogen lattices whose atoms are aligned on top of each other. Seen from above, they form a honeycomb lattice.

TMDCs were known and studied as a bulk material since more than five decades [30]. The more recent possibility to do experiments on freestanding monolayers [8] has brought these materials back into focus, as for instance bulk TaS_2 shows superconductivity [31] and formation of charge density waves [32], so the effect of the reduction of dimensionality on these phenomena can be studied on it [1, 33].

III.2.1 Charge-density waves

A charge density wave is a periodic modulation of the electronic charge density of a solid. This changes the potential acting on the nuclei, so a distortion of the lattice accompanies the formation of a charge-density wave, to the point where both terms are used interchangeably. Fig. III.2 shows a charge-density wave in 2H-TaS_2.

A simple model for the formation of charge-density waves in a one-dimensional case was given by Peierls in 1955. Following the review by Grüner [34], the argument is as follows: the

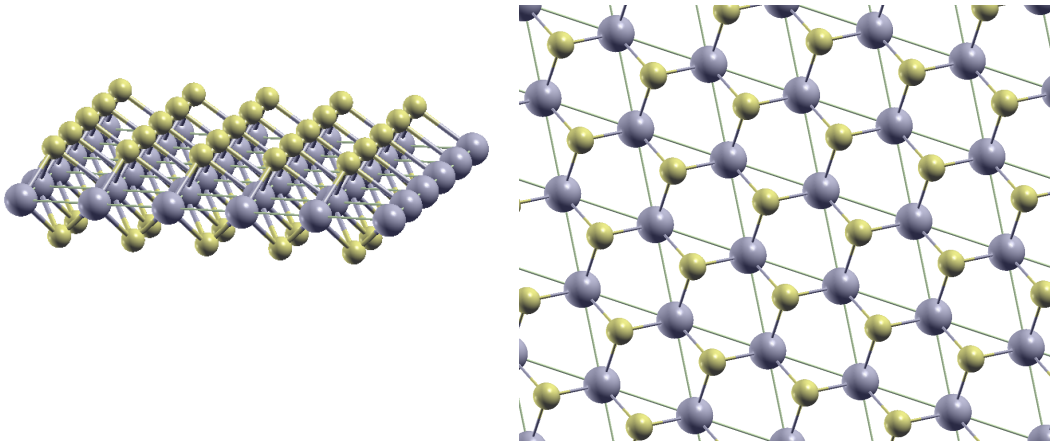


Figure III.1: *Crystal structure of a 2H-TaS_2 monolayer as seen from the side (left) and from the top (right). Visualized using XCrySDen [29]*

formation of a periodic distortion of the lattice creates a unit cell twice as large as the original one, so the Brillouin zone becomes half as large. Thus, the bands fold back onto this smaller Brillouin zone and split due to an avoided crossing. This results in a gap at the Fermi level.

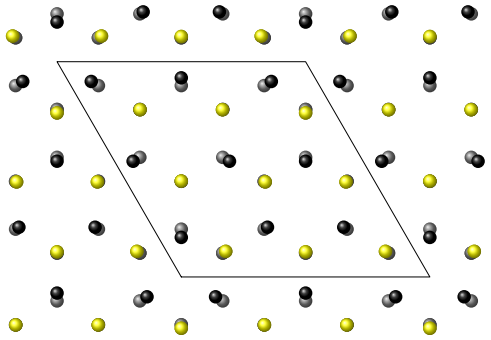


Figure III.2: *TaS_2 charge density wave. Gray dots are atoms in the symmetric phase, yellow/black dots are the Tantalum/Sulfur atoms in the charge density wave phase.*

III.2.2 Computational parameters

All calculations on the 2H-TaS_2 charge-density wave were made with a plane-wave cutoff of 100 Ry and on a 12×12 k-point grid. The calculations use a PBE XC-functional with a

norm-conserving PP generated by Hartwigsen et al. [35]. Input files for both the symmetric and charge density wave phase were kindly provided by Dr. Jan Berges.

IV Parallelization of electronic-structure calculations

The `PWscf` (Plane-Wave Self-Consistent Field) package is one of the core modules of Quantum ESPRESSO, as many other modules require ground state density and total energy as input. This chapter deals with examining the best ways to run `PWscf` calculations in the `scf` mode. All benchmarks except when otherwise noted are averaged over 10 runs.

IV.1 First scaling tests

The first step in analyzing the scaling of the `PWscf` module is to perform a baseline scaling test without any optimizations applied. In Fig. IV.1 to IV.4 two scaling tests on the earlier mentioned benchmarking systems Si and TaS_2 are depicted. The tests are run using Quantum ESPRESSO 7.0, compiled using the Fortran and C compilers in OpenMPI 4.1.0, without any of the compilation or runtime optimization parameters mentioned in section II.2 used.

As discussed in sec. II.2.3, three different metrics of scalability can be deduced from the time data given by Quantum ESPRESSO:

- runtime: absolute runtime (walltime) of the compute job
- speedup: runtime divided by runtime of the job on a single core
- wait time: percentage of wall time used by system tasks, e.g. writing to disk, etc.

These are depicted in fig. IV.1 and IV.2 for the silicon benchmarking system.

On a single node, the speedup does scale linearly with the number of processors until around 10 processors, but with a slope of 1/2 instead of 1 (which would mean ideal scaling). Beyond this number, the slope decreases even more such that a maximal speedup of around 7 is achieved for 20 processors used. One compute node is equipped with 20 cores. Hence trying to scale the communication intensive calculations beyond that threshold makes the calculations run even slower than on a single core. Interestingly, the wait time plot in fig. IV.2 shows that 10 % to 40 % of runtime is taken by wait time already for less than 20 processors. As discussed in sec. II.1, this is a sign of poor parallelization, which can explain the poor scaling seen in fig. IV.1.

Fig. IV.3 and IV.4 show the same scaling test run for the TaS_2 benchmarking system. Here, the speedup is not taken as overall runtime divided by runtime on a single core, as the memory required is more than what can be accessed by a single core. Instead, an estimate of the single core runtime is made by multiplying the runtime of the job on 4 cores by 4. This assumes perfect scaling for 1-4 processors, but the relative scaling is accurate, no matter the accuracy of this assumption.

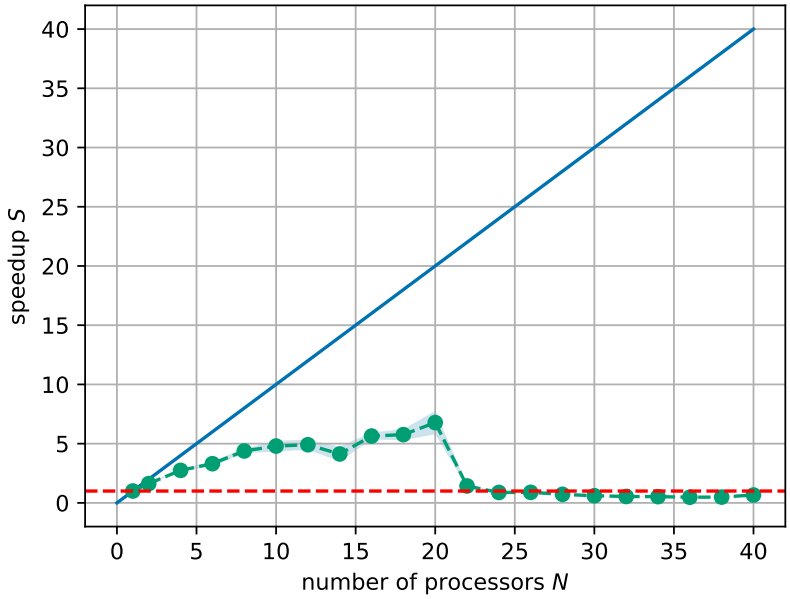


Figure IV.1: *Scalability for the Si benchmarking system. The speedup shows linear scaling with a slope of 1 on one node, with worse than single core performance on more than on node.* Quantum ESPRESSO 7.0 compiled with OpenMPI 4.1.0, `nk 1` and `nd 1`

The scaling test on the TaS_2 system shows much better scaling. For up to 20 processors, the speedup follows the ideal scaling with a stark decline with more processors. This is also reflected

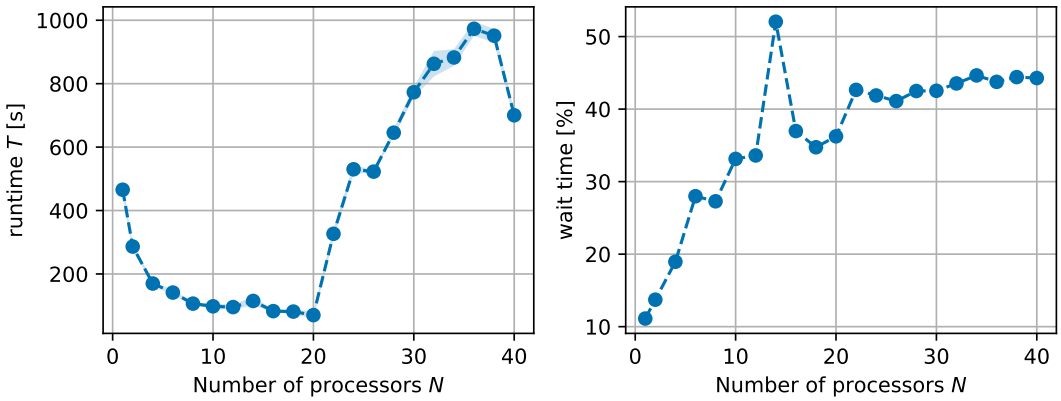


Figure IV.2: *Absolute runtime and wait time for the scalability test on the Si benchmarking system. The runtime shows the scale of how execution on a single node versus two nodes slows down the calculation, the wait time shows that when adding more processors, a higher percentage of runtime is spent in tasks not relevant to the calculation.* Quantum ESPRESSO compiled with OpenMPI 4.1.0, `nk 1` and `nd 1`.

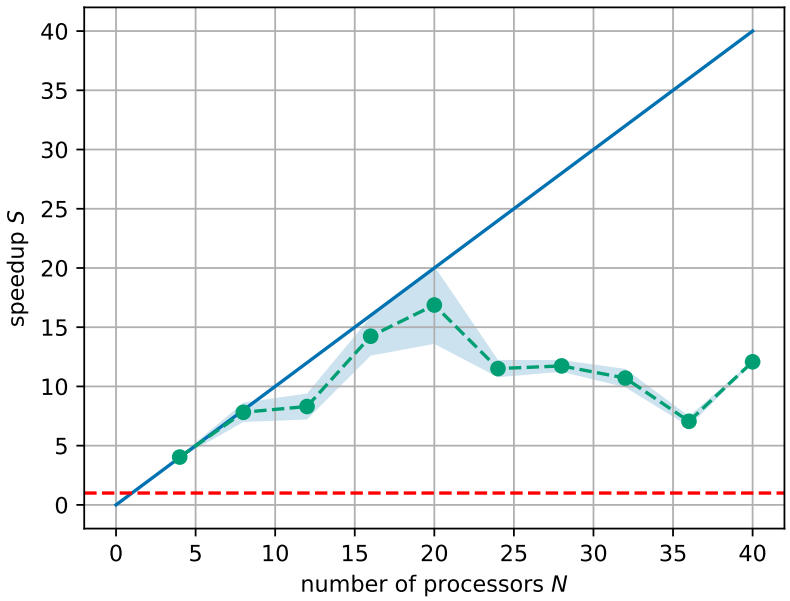


Figure IV.3: *Scalability for the TaS_2 benchmarking system. The speedup is linear with N on a single node, with a drop in speedup over two nodes.* QUANTUM ESPRESSO 7.0 compiled with OpenMPI 4.1.0, `nk 1` and `nd 1`

in the wait time in fig. IV.4, as it goes from a small constant value for under 20 processors

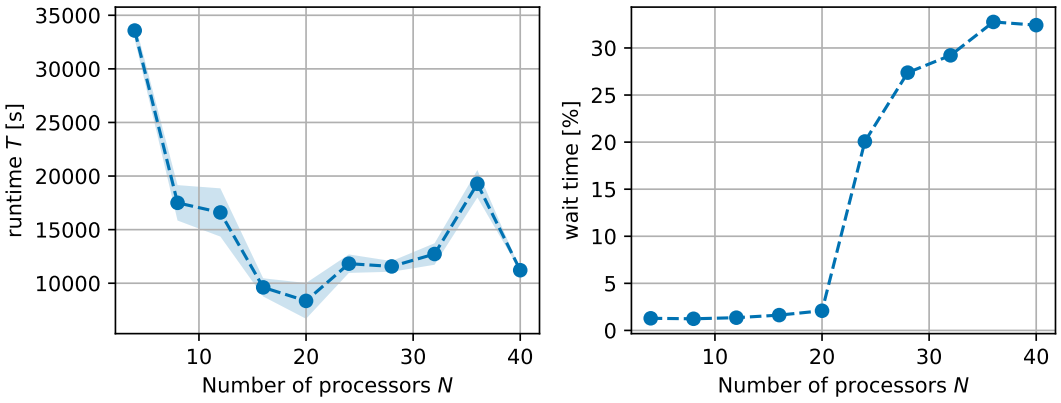


Figure IV.4: *Absolute runtime and wait time for the scalability test on the TaS_2 benchmarking system. The difference between execution on a single node and two nodes is seen in the longer runtime and higher percentage of wait time.* QUANTUM ESPRESSO 7.0 compiled with OpenMPI 4.1.0, `nk 1` and `nd 1`

to some kind of dependence on the number of processors, which hints to communication or bottlenecks being a limiting factor here.

The comparison between the silicon and TaS_2 benchmarks points towards better parallelizeability for system with more electrons and by extension bigger matrices and longer iteration times. As such they profit more from using more processors than systems with just a few electrons.

These scaling tests now pose the question, how better scaling over more than one node can be achieved.

IV.2 Testing different compilers and mathematical libraries

A first strategy for solving issues with parallelization is trying different compilers and mathematical libraries. As discussed in sec. II.2.1, Quantum ESPRESSO can make use of a variety of software packages available on the PHYSnet cluster. The benchmarks in fig. IV.5 are run with the following software combinations:

(a) OpenMPI 4.1.0 and Quantum ESPRESSO provided BLAS/LAPACK, so the baseline test discussed in sec. IV.1

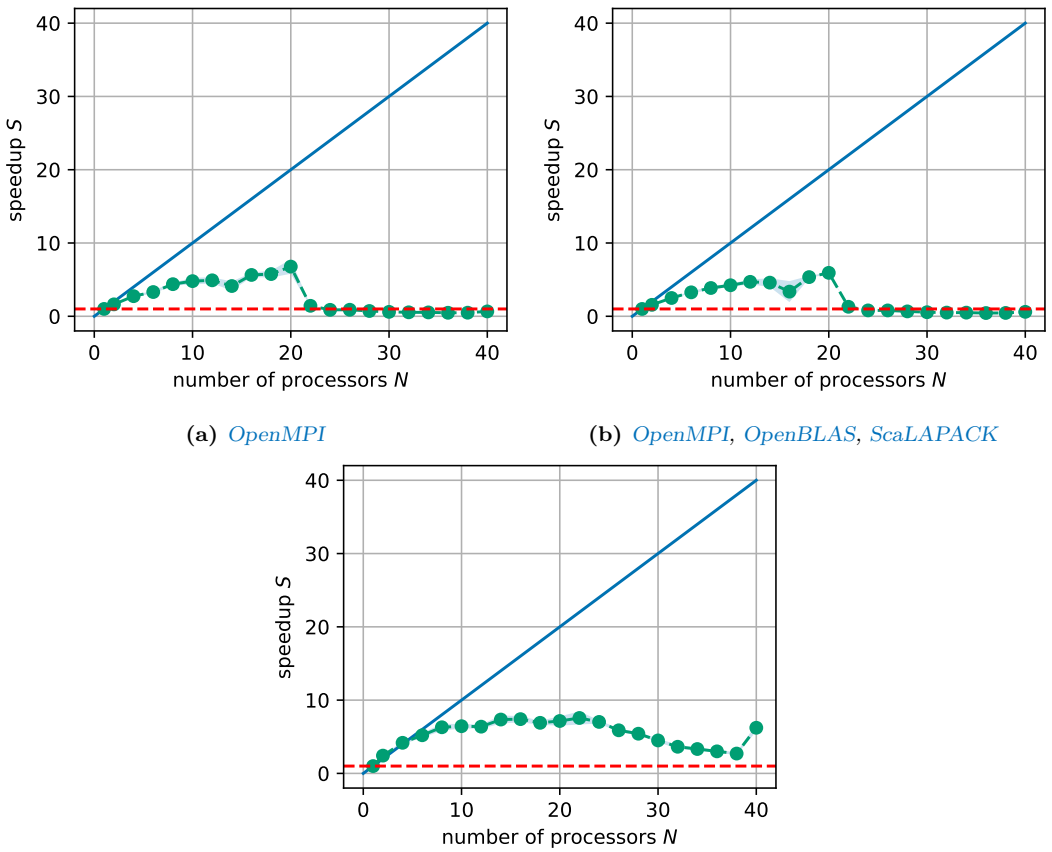
(b) OpenMPI 4.1.0, OpenBLAS 0.3.20 and ScaLAPACK 2.2.0

(c) Intel oneAPI 2021.4

Fig. IV.5 shows that just using another BLAS/LAPACK library (OpenBLAS in this case) with the same MPI version does not change the scaling behavior, in contrast to using Intels Intel oneAPI packages. The latter shows optimal scaling behavior for up to 6 processors. It is however important to also look at the total runtime in this context.

Fig. IV.6 shows the absolute runtime for both the OpenMPI and Intel oneAPI benchmarks. This explains the difference in scaling seen in the speedup plots: the runtime on a single core is significantly higher for the Intel oneAPI benchmark, so even though the runtime between both benchmarks is about the same starting from around 10 processors there is a difference in speedup. To assess this more quantitatively, tab. IV.1 lists the average runtime for some selected number of processors. Importantly, the runtime for the Intel oneAPI benchmark is faster for smaller numbers of processors (except 1), yet only 15 % for 2 cores and even smaller differences for more cores, with the OpenMPI calculation being even a little faster for 20 processors again.

The same benchmark with the Intel oneAPI compiled version of Quantum ESPRESSO is shown in fig. IV.7 and IV.8 for TaS_2. For this system, the speedup roughly follows Amdahl's law, discussed in sec. II.1 with a linear growth in speedup up to 32 processors with a saturation and only a small gain in speedup with more processors. In contrast to the benchmark with just OpenMPI (fig. IV.3) there is no drop in speedup after 20 processors. This is remarkable and also a difference to the silicon benchmarking system, where 1 node is a definite upper bound for scalability. An explanation for this behavior can be made with the help of Amdahl's law again. As discussed in sec. II.1, the exact form of the speedup is not dependent on absolute times of parallelized and unparallelized parts of a calculation, but rather the proportion between these two (and can thus be characterized by just the purely serial part s). This means that



(a) *OpenMPI*

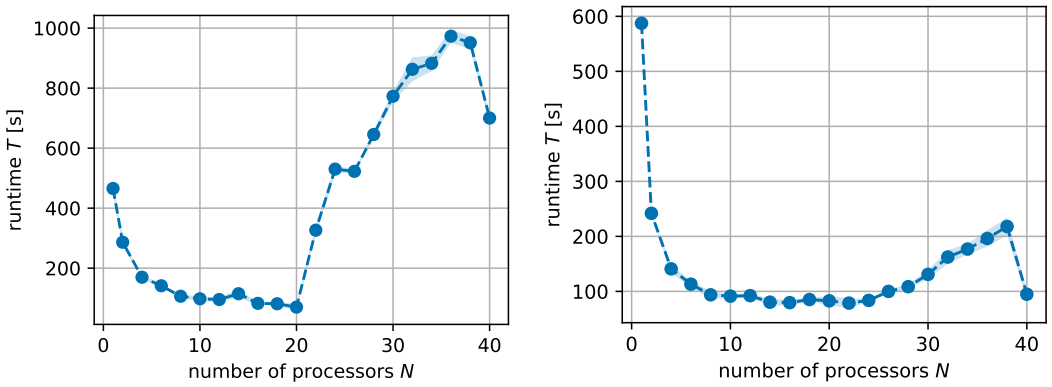
(b) *OpenMPI*, *OpenBLAS*, *ScaLAPACK*

(c) *Intel oneAPI*

Figure IV.5: *Scalability for the Si benchmarking system with different combinations of compilers and mathematical libraries. Intel oneAPI compilers show a different scaling behavior, with better scalability across multiple nodes.* `nk 1` and `nd 1`

for a more expensive system such as the TaS_2 benchmarking system, when the absolute time for communication, data distribution and collection stays roughly the same and the time for a single KS iteration (which can be parallelized) is way longer, the proportion of the purely serial part s gets smaller and the scaling behavior changes significantly.

Moreover, the absolute runtime shown in fig. IV.8 shows that the calculations not only scale better than with OpenMPI, but they are significantly faster: whereas the minimum for the OpenMPI benchmark is around 1 h50 min for 20 processors, the Intel oneAPI benchmark averages around 40 min for 24 processors. The wait time across the whole range of processors is significantly lower than in the OpenMPI benchmarks. This speaks for generally better parallelization, which confirms the observations made on the speedup measured.



(a) *OpenMPI 4.1.0* **(b)** *Intel oneAPI 2021.4*

Figure IV.6: *Comparison of absolute runtimes between* Quantum ESPRESSO *compiled with OpenMPI and Intel oneAPI for the Si benchmarking system. The runtimes show again the different scaling behavior, while minimal absolute runtime is the same for both.* `nk 1` and `nd 1`

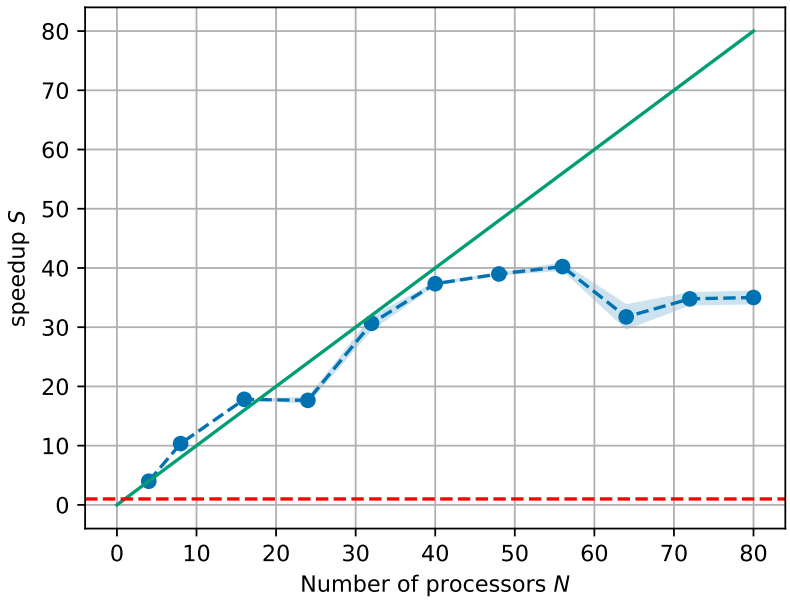


Figure IV.7: *Scalability for the TaS_2 benchmarking system. The scaling is close to optimal for up around 40 processors.* Quantum ESPRESSO 7.0 compiled with Intel oneAPI 2021.4, `nk 1` and `nd 1`

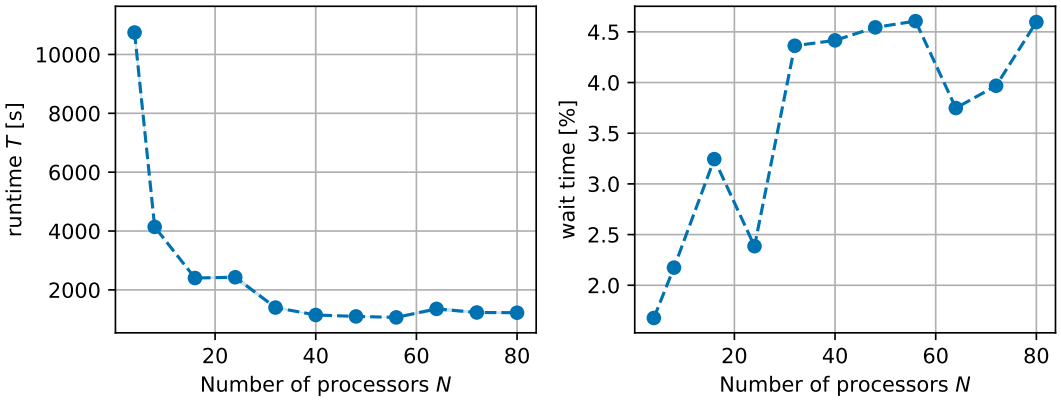


Figure IV.8: *Absolute runtime and wait time for the scalability test on the TaS_2 benchmarking system. The wait time is under 5 % across the whole range of processors.* Quantum ESPRESSO compiled with Intel oneAPI 2021.4, `nk 1` and `nd 1`

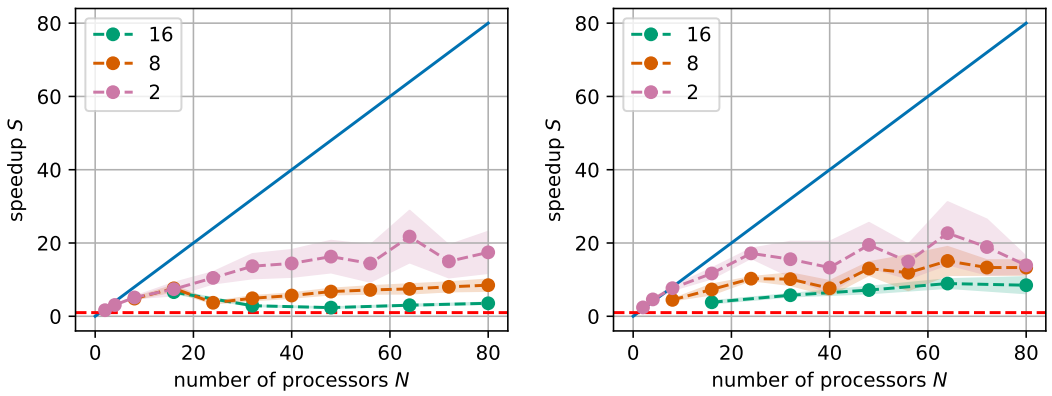
While the benchmarks on the silicon system do not seem to favor one set of compilers over the other, the tests on the TaS_2 system clearly show the advantages of using Intel oneAPI on the Intel hardware in the PHYSnet cluster.

The observations on how many processors are optimal for certain kinds of systems not only stand for themselves as a statement about scaling on a single node or a small number of nodes, but also provide a basis for scaling beyond the respective optimal ranges of processors for both systems: The k-point parallelization explained in sec. II.2.2 can distribute the workload in such a way that processor pools of sizes within this range work on individual k points and as such can provide optimal scaling within one pool while also not losing performance because the pools do not need to communicate with each other in the same order of magnitude as the pools have to communicate within themselves.

Keeping the results of this section in mind, an estimate for the quality of k-point parallelization can already be made: For the silicon system, the size of pools should not be bigger than 6 processors for optimal scaling and for the TaS_2 system they should not be bigger than 32 processors.

Table IV.1: *Selected absolute runtimes of Quantum ESPRESSO compiled with OpenMPI 4.1.0 and Intel oneAPI 2021.4 for the Si benchmarking system,* `nk 1` and `nd 1`

Number of processors	OpenMPI	Intel oneAPI
1	466 s	587 s
2	286 s	242 s
4	170 s	141 s
10	97.9 s	91.3 s
20	70.2 s	82.8 s



(a) *OpenMPI 4.1.0* **(b)** *Intel oneAPI 2021.4*

Figure IV.9: *Scalability utilizing k-point parallelization for the Si benchmarking system with 3 different sizes of processor pools. The size is determined by the parameter* `nk` *via* size of pools = number of processors / nk. *The maximal speedup is double the speedup reached in benchmarks without k-point parallelization.*

IV.3 Using the parallelization parameters of Quantum ESPRESSO

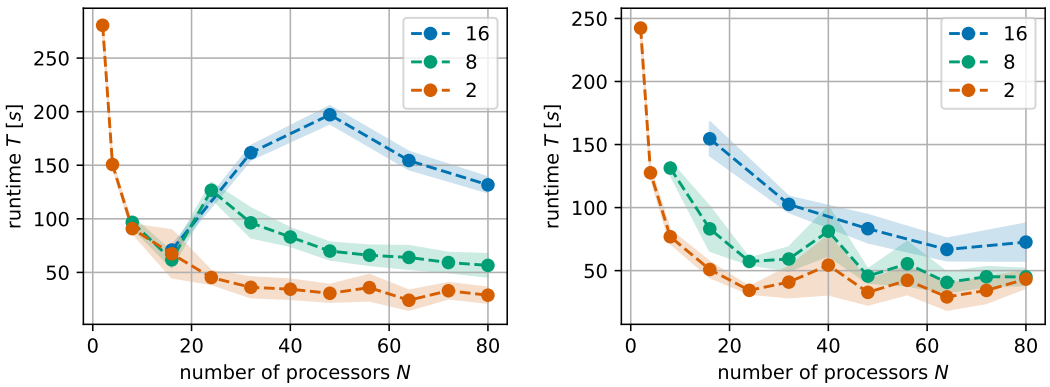
As detailed in section II.2.2, Quantum ESPRESSO offers ways to manage how the workload is distributed among the processors. In `pw.x` the default plane-wave parallelization, k-point parallelization and linear-algebra parallelization are implemented. While plane-wave parallelization is automatically applied, k-point and linear-algebra parallelization can be controlled and will be tested in this section.

IV.3.1 k point parallelization

The benchmark shown in fig. IV.9 is set up as follows: for a given number of processors N_p, the parameter N_k splits the N_p processors into N_k processors pools. As the number of processors in one pool has to be a whole number, only certain combinations of N_p and N_k are possible, for example $N_p = 32$ could be split into processor pools of size 2 with $N_k = 16$, size 8 with $N_k = 4$ or size 16 with $N_k = 2$. This leads to choosing the size of the processor pools as a variable, not the parameter `nk`.

Fig. IV.9 shows the scaling for poolsizes 2, 8 and 16 for Quantum ESPRESSO being compiled with OpenMPI and Intel oneAPI.

The speedup depicted in fig. IV.9 shows that using k-point parallelization with a pool size of 2 improves the scaling behavior not only on one node, but especially over more than one node. While the speedup without k-point parallelization hits a plateau when using more than 6 processors, k-point parallelization enables scaling up until 24 processors. The bigger pool



(a) *OpenMPI 4.1.0* **(b)** *Intel oneAPI 2021.4*

Figure IV.10: *Absolute runtime for the scalability test with k-point parallelization for the Si benchmarking system with 3 different sizes of processor pools. The runtimes show different scaling behavior depending on the pool size, relevant is the best scaling pool size 2. Here, minimal runtimes differ only by a small amount between OpenMPI and Intel oneAPI.* `nd 1`

sizes do not scale as well, which is in agreement with the results of the benchmarks without k-point parallelization presented in the previous section.

The runtimes depicted in fig. IV.10 show that the choice between using OpenMPI and Intel oneAPI has an effect on how the total execution time scales with the number of processors especially for the bigger pool sizes, but the relevant, best scaling pool size 2 case shows no difference in execution time between the two benchmarks. This follows the observations made in the benchmarks without k-point parallelization.

For both benchmarks, the ideal number of processors seems to be around 24, after that the runtime is subject to significant fluctuations and does not decrease in a predictable way. The average runtime for 24 processors are 45.2 s (OpenMPI) and 34.4 s (Intel oneAPI) respectively, so about a 10 seconds difference and also around half the minimal time (70.2 s and 82.8 s at 20 processors) in comparison to the benchmark without k-point parallelization.

The same scaling test is applied to the TaS_2 benchmarking system in fig. IV.11 and IV.12.

Remarkably, the scaling behavior is swapped in comparison to fig. IV.9, as the pool size 2 saturates and the bigger pool sizes show way better scaling behavior. As already alluded to in sec. IV.2, the calculations on the TaS_2 system profit more from parallelization and as such scale better for bigger pool sizes up until 36 processors in one pool, which is around the upper limit established in the benchmark without k-point parallelization.

The minimal runtime achieved with k-point parallelization is about 5 min for pool size 36, with the other pool sizes (except 2) being 1-3 min slower. Fig. IV.12 shows a distribution of wait times between about 4 % and 6 % of the full wall time, with a slight overall increase when going over 160 processors. This suggests very good parallelization across the whole range of processors.

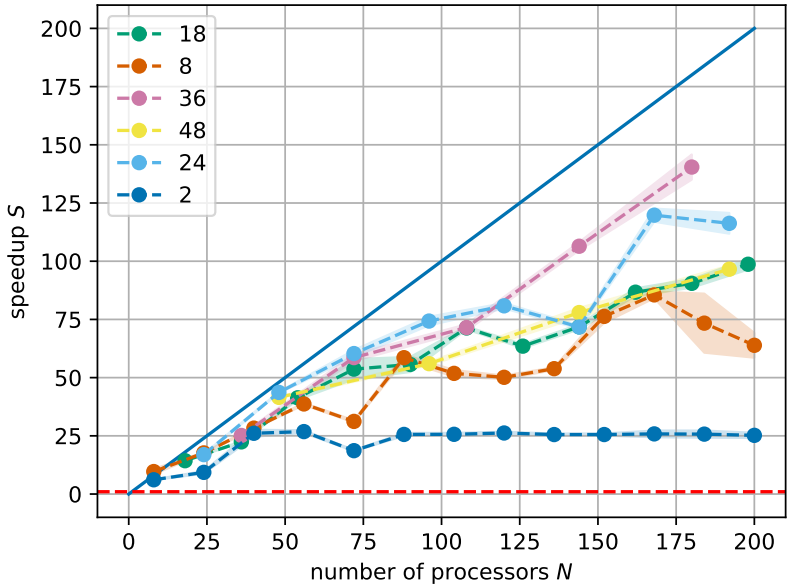


Figure IV.11: *Scalability utilizing k-point parallelization for the TaS_2 benchmarking system over a range of processor pools. The speedup actually increases over the upper bound of 40 processors without k-point parallelization, with pool size 36 parallelizing best.* Quantum ESPRESSO compiled with Intel oneAPI 2021.4, `nd 1`

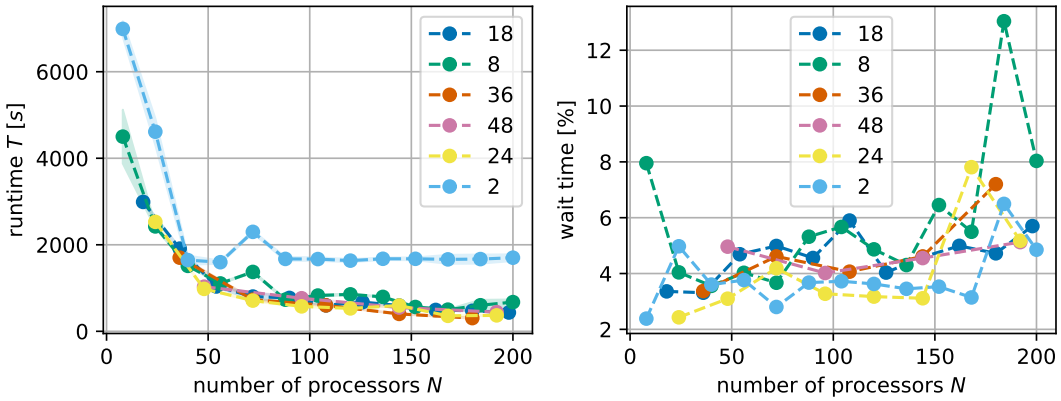


Figure IV.12: *Absolute runtime and wait time for the scalability test with k-point parallelization for the TaS_2 benchmarking system over a range of processor pools. The wait time is under 10 % over almost the whole range of processors.* Quantum ESPRESSO compiled with Intel oneAPI 2021.4, `nd 1`

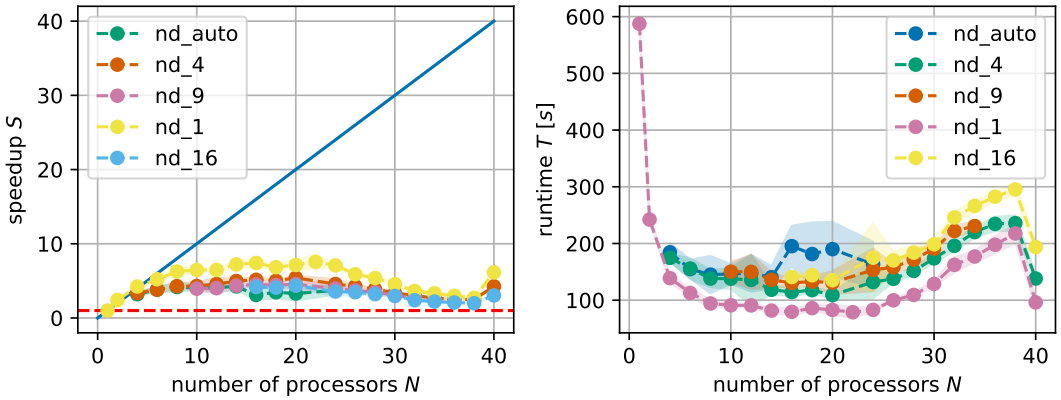


Figure IV.13: *Scalability and runtime utilizing linear-algebra parallelization for the Si benchmarking system over a range of values for the parameter* `nd`*. Using linear-algebra group size slows down the calculations in comparison to* `nd 1`*.* QUANTUM ESPRESSO compiled with Intel oneAPI 2021.4, `nk 1`

IV.3.2 Linear-algebra parallelization

Fig. IV.13 shows the scaling behavior for different values of the parameter `nd`. Here, nd_auto means that no value for `nd` is specified and QUANTUM ESPRESSO automatically chooses the biggest square number smaller than the number of processors. It is clearly shown that using linear-algebra parallelization slows the calculation down significantly for the silicon system.

Interestingly, this again is not reproduced for the more expensive TaS_2 benchmarking system. Fig. IV.14 shows consistent times across all values for `nd`.

Those results are already hinted at in the `PWscf` user guide [25]. Here, in the guide for choosing parallelization parameters, using linear-algebra parallelization is recommended when the number of KS states is a few hundred or more. The silicon system has 8 electrons and is as such described with 4 KS states, the TaS_2 system has 153 electrons, so QUANTUM ESPRESSO uses 92 KS states. In case of metallic materials, the band occupation is smeared around the Fermi energy to avoid level crossings, so more KS states than $\frac{1}{2} *$ (number of electrons) are needed to account for that. Evidently, this number of KS states is on the edge of linear-algebra parallelization actually speeding up calculations.

IV.4 Comparison with calculations on the HLRN cluster

All calculations so far were exclusively run on the PHYSnet cluster and as such are limited by hardware and configuration present in the cluster. To assess this limitation, the k-point benchmarks from sec. IV.3.1 were run again on another cluster, the HLRN cluster in particular. The North German Supercomputing Alliance (Norddeutscher Verbund für Hoch- und Höchstleistungsrechnen - HLRN) operates a distributed supercomputer system at the Georg-August-Universität Göttingen and the Zuse Institute Berlin. The current iteration

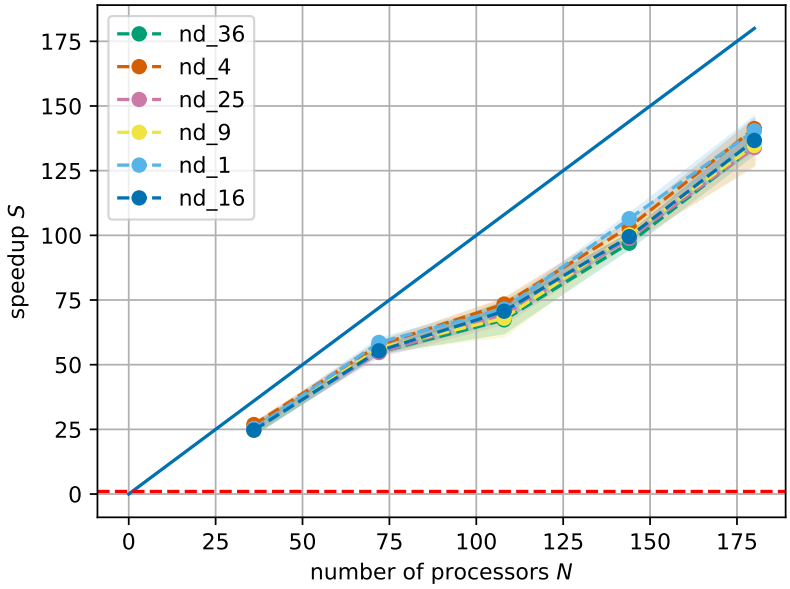


Figure IV.14: *Scalability utilizing linear-algebra parallelization for the TaS_2 benchmarking system over a range of values for the parameter* `nd`*. Speedup is the same for every linear-algebra group size.* QUANTUM ESPRESSO compiled with Intel oneAPI 2021.4, `nk` chosen such that pool size = 36

HLRN-IV has nodes with 2 Intel Cascade Lake Platinum 9242 CPUs (48 cores each) and an Omni-Path (Intels proprietary Infiniband competitor) connection between nodes. QUANTUM ESPRESSO is compiled with Intel Parallel Studio XE Composer Edition 2019 Update 5 (which is the predecessor of Intel oneAPI, bundled without MPI on the cluster) and Intel MPI 2018.5.

Fig. IV.15 and IV.16 show the benchmarks for the silicon system.

The scaling behavior has some striking differences in comparison with the same benchmarks run on the PHYSnet cluster. First of all, speedup and runtimes are very consistent across runs, with only a minimal variance across the whole range of processors. On a single node this is similar to the results from the benchmarks run on the PHYSnet, but on the HLRN cluster this also holds true across two nodes (for more than 96 processors). This is most likely due to the HLRN cluster being equipped with better communication hardware, which is also the reason for the speedup further increasing over two nodes, whereas in the benchmark on the PHYSnet cluster the maximum speedup of around 20 was already achieved for 24 processors.

Another difference lies in the fact that the pool sizes 2 and 8 show a similar speedup with pool size 16 being a bit worse.

This fact shows that while recommendations for parallelization parameters can be qualitatively made based on system size, the optimal size and processor range can vary depending on the compute cluster the calculations are run on.

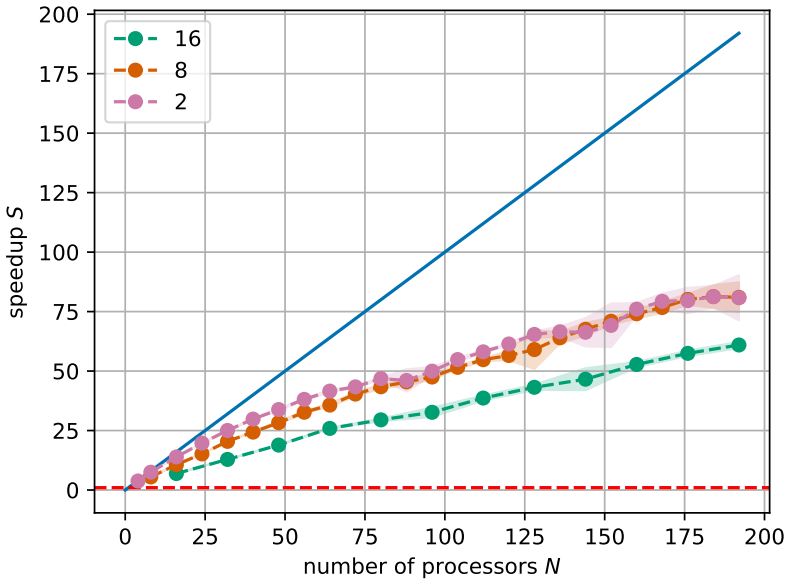


Figure IV.15: *Scalability utilizing k-point parallelization for the Si benchmarking system with 3 different sizes of processor pools run on the HLRN cluster. Across the whole range of processors, calculations get faster when adding more processors.* QUANTUM ESPRESSO compiled with Intel Parallel Studio 2019u5, Intel MPI 2018.5, `nd 1`

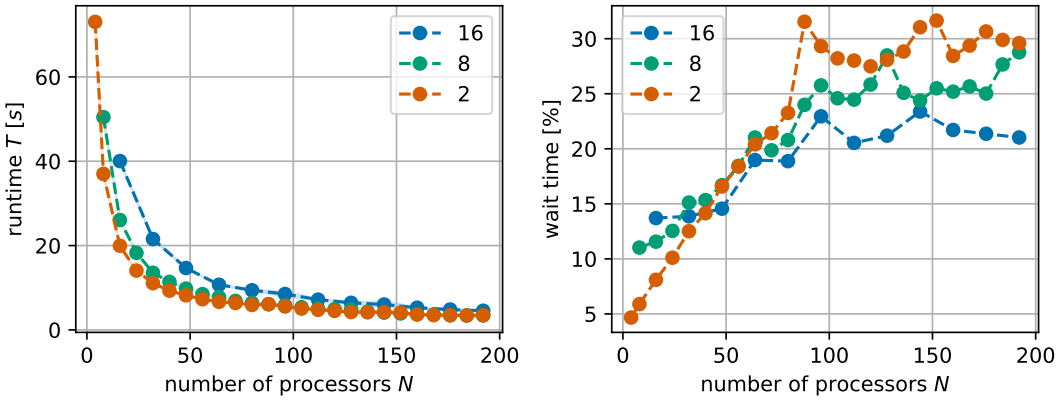


Figure IV.16: *Absolute runtime and wait time for the scalability test with k-point parallelization for the Si benchmarking system run on the HLRN cluster. The wait time shows a different behavior between running on one and two nodes.* QUANTUM ESPRESSO compiled with Intel Parallel Studio 2019u5, Intel MPI 2018.5, `nd 1`

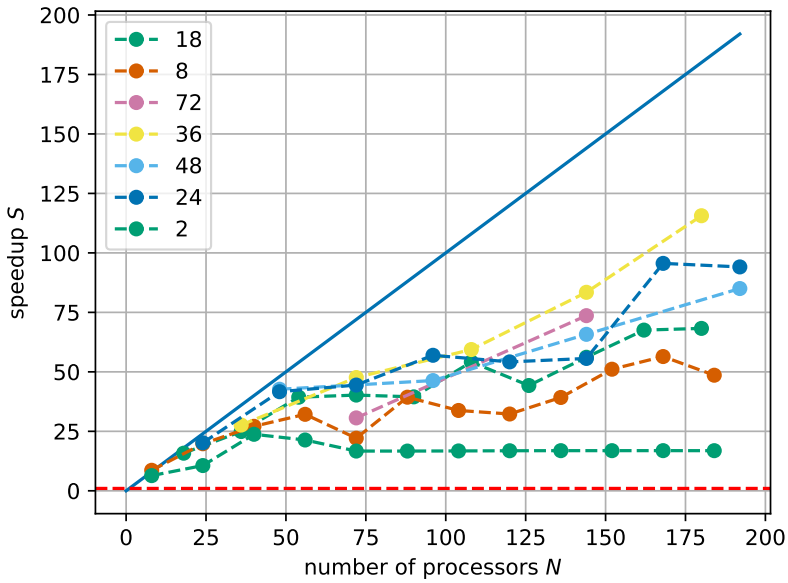


Figure IV.17: *Scalability utilizing k-point parallelization for the TaS_2 benchmarking system over a range of processor-pool sizes run on the HLRN cluster. Pool size 36 shows the best scaling behavior.* QUANTUM ESPRESSO compiled with Intel Parallel Studio 2019u5, Intel MPI 2018.5, `nd 1`

Fig. IV.17 and IV.18 show the benchmarks for the TaS_2 system. This was only run a single time, not averaged over multiple runs.

Interestingly, the upper bound for pool size is the same as for the calculations on the PHYSnet. In both cases, bigger pool sizes perform better up to 36 processors, with more processors per pool not showing better scaling. This suggests that this limit is not dependent on the compute cluster, but instead a property of the TaS_2 system: Just looking at the PHYSnet cluster, 36 as an upper limit might allude to an interpretation involving the size of nodes, where 36 processors is just under 2 full nodes and everything more will be spread over at least 3 nodes, with possibly more communication slowing down calculations. On the HLRN, the processor counts per CPU and node are different, so an analogous interpretation would suggest that the scaling gets better for pool sizes up to some multiple of 48 (the number of processors on one CPU). This is not the case, the processor pool of size 48 already shows worse scaling than the size 36.

This is an important result, as for the more expensive TaS_2 system the topology of the compute cluster seems to not be important for the scaling behavior. This opens up the possibility of using the PHYSnet not just for calculations but also for first finding the best parameters for a particular system (so just running a few iterations) and then doing the real calculation on a cluster like the HLRN.

While the speedup plots look very similar between the PHYSnet and the HLRN, the absolute runtime in fig. IV.18 shows a massive difference between the two clusters. While the minimal

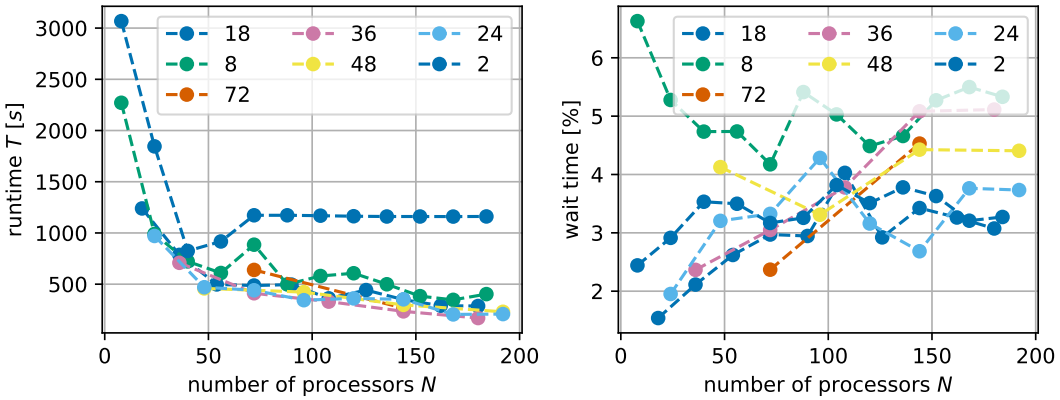


Figure IV.18: *Absolute runtime and wait time for the scalability test with k-point parallelization for the TaS_2 benchmarking system run on the HLRN cluster. The wait time is in a range of around 2-6 % across all pool sizes and processors.* Quantum ESPRESSO compiled with Intel Parallel Studio 2019u5, Intel MPI 2018.5, `nd 1`

runtime achieved on the PHYSnet is about 5 min, the minimal time needed on the HLRN cluster is about 2 min45 s, so a time difference of factor 2.

IV.5 Conclusion: Parameters for optimal scaling

The benchmarks testing Quantum ESPRESSO's `PWscf` module showed that the speedup gained from parallelization is highly dependent on the specific system. While calculations on the relatively inexpensive silicon benchmarking system gained performance just in a range up to 24 processors and were faster by a factor of 2, the more expensive TaS_2 benchmarking system benefitted more from parallelization itself and as such from all efforts improving parallelization.

Using the compilers and mathematical libraries from the Intel oneAPI package was shown to improve scalability beyond a single node, with the maximum speedup at around 32 processors for the TaS_2 benchmarking system. This value was then confirmed to be the best choice for the size of processor pools in k-point parallelization not just on the PHYSnet cluster, but also on the more capable HLRN cluster. Using linear-algebra parallelization had a negative impact on calculations on the silicon system and no effect for calculations on the TaS_2 benchmarking system.

A generalization for other systems beyond a qualitatively assessment that more expensive system profit more from parallelization than smaller system is not possible, but this chapter gives a guide on how to examine a system in order to get the optimal parameters.

V Parallelization of phonon calculations

The `PHonon` package enables calculations of phonon frequencies and eigenmodes. This chapter examines the best ways to run `PHonon` calculations. The benchmarks are averaged over 10 runs.

V.1 Optimal parallelization parameters for phonon calculations

As discussed in sec. II.2.2, the `PHonon` package offers the same three parallelization levels as the `PWscf` package, namely plane-wave, k-point and linear-algebra parallelization. Furthermore, parallelization on q-points (so called image parallelization) can be used.

V.1.1 k point parallelization

In a first step, the same k-point parallelization benchmark as in sec. IV.3.1 is run. This is depicted in fig. V.1.

Interestingly, the result from the `PWscf` calculation on silicon from sec. IV.3.1 is not reproduced here: the smallest pool size of 2 is not the one parallelizing best, but instead it is pool size 8. Furthermore, for more than 50 processors, even the biggest pool size 18 shows better scaling than the pool size 2. The general picture is similar to `PWscf` benchmark with k-point parallelization on the TaS_2 benchmarking system in sec. IV.3.1, as there is no optimal pool size over the whole range of processors, instead some pool sizes seem to work best for some ranges.

One possible conclusion to draw from the benchmarks in chapter IV is, that in general longer runtime results in the calculation profiting more from parallelization and as such also from bigger pool sizes. The phonon benchmark has a similar runtime to the `PWscf` benchmark on TaS_2 shown in sec. IV.3.1, so following this argumentation a similar scaling behavior is expected. Comparison in wait time reveals the differences in the quality of parallelization between the two systems, which results in the observed different scaling. Whereas the `PWscf` benchmark on TaS_2 had wait time not exceeding about 8 % of the wall time, the wait time shown in fig. V.2 between 10 % and 50 %.

A possible explanation for these differences between the two kinds of calculation can be found in how the time is actually spent during the calculation (which can be found in the Quantum ESPRESSO output files): In the case of the phonon calculation on silicon, the time of one iteration is on the scale of seconds, whereas one iteration for the `PWscf` calculation on TaS_2 is about 1 min. This means that the proportion of time spent on the distribution of data is

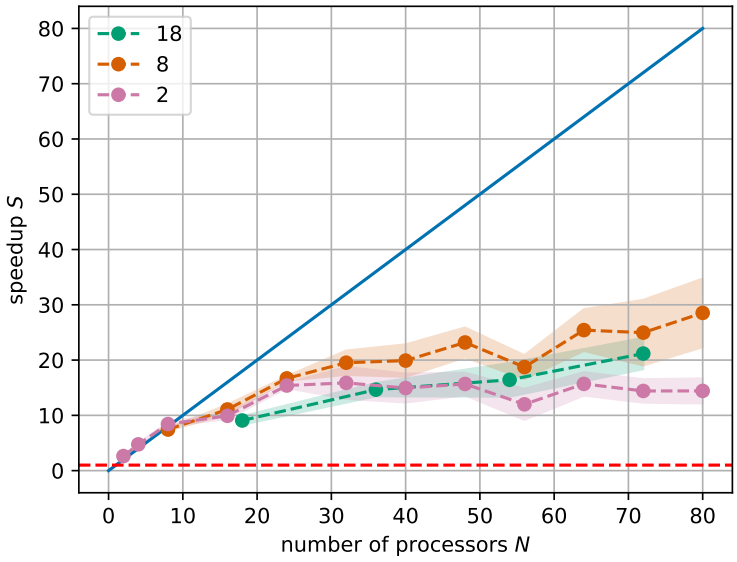


Figure V.1: *Scalability utilizing k-point parallelization for the Si benchmarking system with three sizes of processor pools. Best scaling behavior is seen for pool size 8.* QUANTUM ESPRESSO compiled with Intel oneAPI 2021.4, `nd 1`

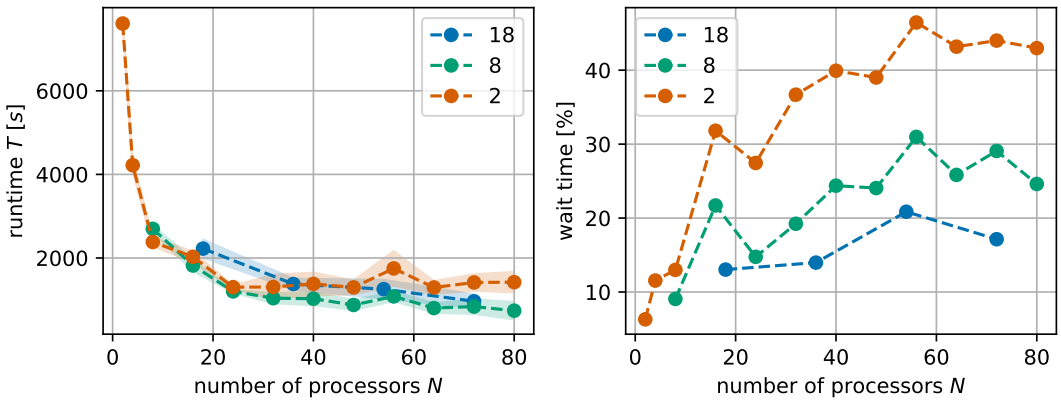


Figure V.2: *Absolute runtime and wait time for the scalability test utilizing k-point parallelization for the Si benchmarking system with three sizes of processor pools. Wait time is highest for the pool size which parallelizes worst, with the other two having lower wait times.* QUANTUM ESPRESSO compiled with Intel oneAPI 2021.4, `nd 1`

bigger for the phonon calculation on silicon compared to the `PWscf` calculation on TaS_2, which introduces wait times.

From the result of this benchmark, the parameters for the image parallelization in sec. V.1.3 can be set: the pool size will be fixed at 8.

V.1.2 Linear-algebra parallelization

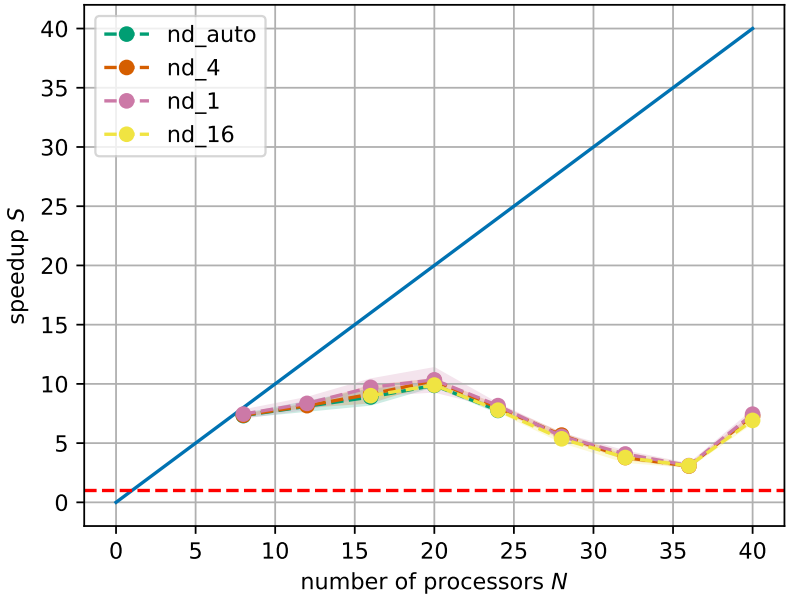


Figure V.3: *Scalability utilizing linear-algebra parallelization for the Si benchmarking system. The scaling is the same across the whole range of processors for every linear-algebra group size.* Quantum ESPRESSO compiled with Intel oneAPI 2021.4, `nk 1`

Fig. V.3 shows that using linear-algebra parallelization has no significant impact on the speedup. This is again in contrast to the `PWscf` results from sec. IV.3, where using linear-algebra slowed down the calculation and more similar to the `PWscf` calculation on TaS_2.

As such, linear-algebra parallelization will not be used in the benchmarks for image parallelization.

V.1.3 Image parallelization

When using image parallelization, Quantum ESPRESSO outputs a separate time report for every image, so one additional step is needed in the analysis: While the total runtime of a calculation is determined by the longest running image, the variation of times between images is important to judge how well the work between images is distributed. This is depicted in fig. V.4.

As the standard deviation between images is just 20-60 s, which amounts to about 10 % of the average runtime, good load balancing between images can be assumed for the silicon benchmarking system.

With the maximum time across images, speedup is then calculated, shown in fig. V.5.

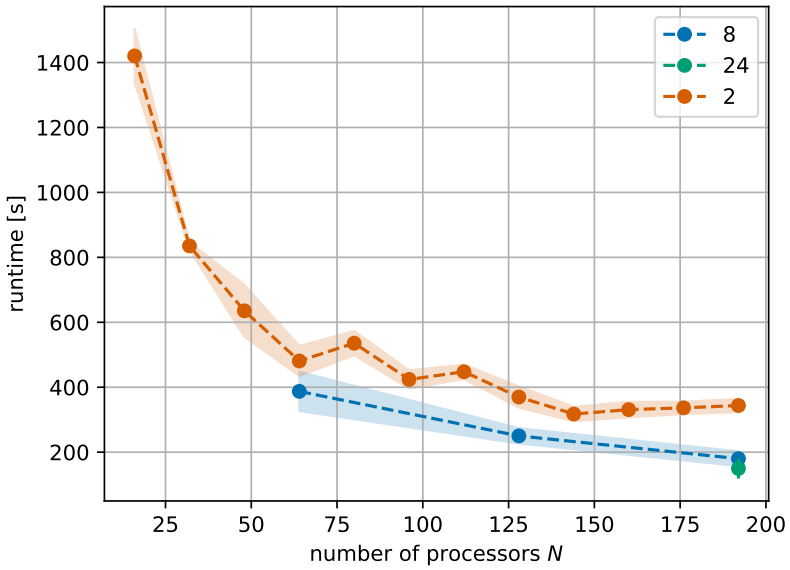


Figure V.4: *Average runtime across images for the scalability test utilizing image and k-point parallelization on the Si benchmarking system with three values of* `ni`*. The transparent line shows standard deviation across images, which shows that all images need around the same time for their respective calculations.* QUANTUM ESPRESSO compiled with Intel oneAPI 2021.4, `nk, ni` chosen such that poolsize = 8, `nd 1`

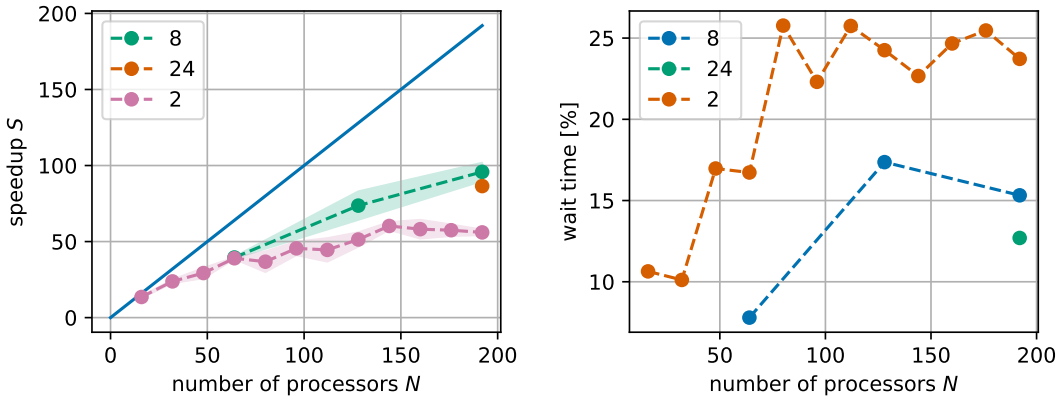


Figure V.5: *Speedup and wait time calculated from the longest running image for the scalability test utilizing image and k-point parallelization on the Si benchmarking system with three values of* `ni`*. Using more images leads to the linear slope of the speedup continuing over more processors. The wait time is lowest for every first data point and lower than without image parallelization in general.* QUANTUM ESPRESSO compiled with Intel oneAPI 2021.4, `nk, ni` chosen such that poolsize = 8, `nd 1`

Table V.1: *Benchmark values for phonon calculations of TaS_2 in the charge density wave phase, run on 180 processors*

	runtime	wait time
pool size 18	3044 min	16 %
pool size 36	2020 min	7.4 %

The speedup shows that using image parallelization helps the phonon calculations scale over more processors than just using k-point parallelization. Even just using 2 images almost doubles the maximal achieved speedup in fig. V.1. This behavior doesn't extend across the whole range of processors, but using more images can partially lift this problem.

The wait time (also calculated from the longest running image) shown in fig. V.2 is particularly good for every first data point, so in cases where only image parallelization without additional k-point parallelization is applied. As an example, for 64 processors, 8 images is ${}^{N_\mathrm{P}}/_{N_i} = 8$, so the pool size is already 8 as required and the parameter `nk` is chosen to be 1. So at least for the silicon system it seems to be advisable to choose the parameter `ni` as large as possible, so that `nk` can be chosen as small as possible while keeping a pool size of 8.

V.2 Phonon calculations on TaS_2

The results from the last section can be used to estimate good parallelization parameters for a phonon calculation at the Γ point for TaS_2 in the charge density wave phase. The calculations were run on 180 processors, once with the previous established optimal pool size of 36 and once with a pool size of 18 for comparison. The relevant benchmark values for this calculation are listed in tab. V.1.

In this calculation the need for a good choice of parallelization parameters becomes especially clear: while the runs share an identical number of processors, the different choice of the parameter `nk` results in a runtime saving of 17 h.

V.3 Conclusion: Parameters for optimal scaling

The benchmarks on Quantum ESPRESSO's `PHonon` module showed that calculation on the silicon system profited more from bigger k-point pools than `PWscf` calculations. Linear-algebra parallelization was shown to have no impact on calculations.

Image parallelization in combination with k-point parallelization led to very good scaling across a wide range of processors. Best results were found when the processors were only split up by image parallelization.

Phonon calculations on TaS_2 in the charge density wave phase were then carried out, with the best time reached for a pool size of 36.

VI Phonon mediated tunneling into TaS_2

VI.1 Amplitude mode in TaS_2 charge density wave

The results of the phonon calculation at the Γ point carried out in sec. V.2 are 81 phonon modes with respective energies and nucleic displacements. One particular mode has the effect of displacing the atoms in the direction of the symmetric phase, thus changing the amplitude of the charge density wave. This is depicted in fig. VI.1. The amplitude mode has an energy of $E_0 = 10.6\,\mathrm{meV}$.

VI.2 Phonon mediated tunneling into Graphene

VI.2.1 Scanning Tunneling Spectroscopy

The description in this section follows the textbook "Surface Science" by Oura et al. [36].

In an STM setup, a ideally atomically sharp metallic tip is placed close to the probed surface (around $5 - 10\,\text{Å}$), such that electrons can tunnel between the tip and the surface. By applying a bias voltage V between the tip and the surface, a current enabled by the tunneling process

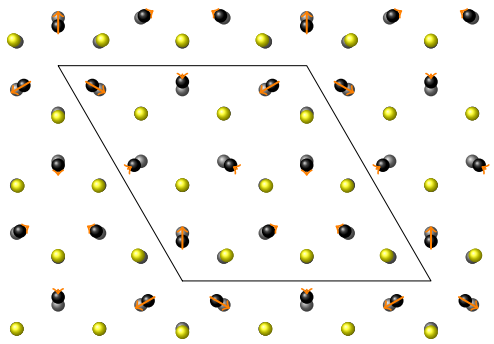


Figure VI.1: *Amplitude mode in TaS_2 charge density wave, visualized from Quantum ESPRESSO calculation. Gray dots are atoms in the symmetric phase, yellow/black dots are the Tantalum/Sulfur atoms in the charge-density-wave phase. The orange arrows show the displacement vectors.*

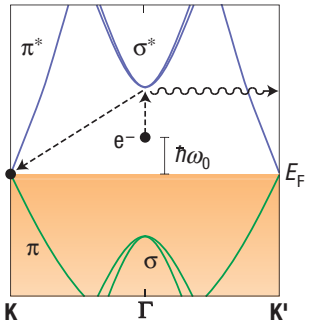


Figure VI.2: *Inelastic tunneling mechanism involving graphene phonon modes near the* $\boldsymbol{K}$ *point in reciprocal space.* Reprinted by permission from Nature Publishing Group: Nature Physics [37], © 2008

flows through the gap. This tunneling current is sharply dependent on the size of the gap, so a extremely high vertical resolution can be achieved. Also due to this fact, 90 % of the tunneling current flows through the the single tip atom closest to the surface, so that very high lateral resolution is also possible. An STM can thus be used to map the topography of a surface with atomic resolution.

When varying the bias voltage V, an STM can also be used to obtain measurements of the density of electronic states in the material, the Scanning Tunneling Spectroscopy (STS) mode of operation. This is due to the tunneling current being determined by summing over electron states in an energy interval defined by the bias voltage. In general, a simple interpretation of STS data is not possible and needs backing in theoretical calculations. This is due to the tunneling current depending on the tunneling matrix element between the state at the tip and the states in the probe, so the tunneling current can be suppressed if the states in the probe don't overlap with the tip.

VI.2.2 Phonon mediated tunneling into Graphene

In a 2008 STS measurement on graphene, a gap feature around the Fermi level was measured [37]. This gap is an example of the problems with interpretation of STS data explained in sec. VI.2.1, as it is not a feature of the density of states of graphene, but a result of the tunneling current being suppressed for the range of bias voltages in the gap.

The underlying mechanism, confirmed with DFT calculations by Wehling et al. [38] is as follows: generally, electrons can elastically tunnel into graphene at the Fermi level near the $\boldsymbol{K}$ point. This elastic process is suppressed because the wave function at the initial state, i.e. the wave functions at the tip, have a momentum distribution centered at $\boldsymbol{k}_{\parallel} = 0$, so the tunneling matrix element is suppressed for large $\boldsymbol{k}$ [39]. For electron energies outside the gap, an inelastic tunneling channel opens, where the electrons first tunnels into the σ^* band and then transitions into an available $\boldsymbol{K}$ point through emission of a $\boldsymbol{K}'$ point phonon (pictured in fig. VI.2).

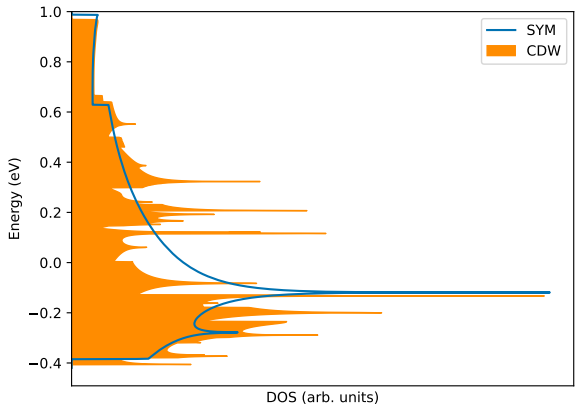


Figure VI.3: *Density of states for TaS_2 in the charge density wave (CDW) and undistorted (SYM) phase. The data was kindly provided by Dr. Jan Berges and has been calculated in Quantum ESPRESSO with the 2D tetrahedron method using 360^2 (1080^2) k points for the CDW (SYM) structure*

VI.3 Phonon mediated tunneling into TaS_2

In a 2019 paper by Hall et al. [1], a similar gap feature with a width of $2\Delta = (32 \pm 9)\,\mathrm{meV}$ was reported in an STS measurement on TaS_2.

This gap is attributed to partial gapping to the formation of the charge density wave, as explained in the one-dimensional case in sec. III.2.

The density of states depicted in fig. VI.3 shows no symmetric gap around the Fermi level, hence the explanation by partial gapping is not confirmed. An alternative explanation can be that the same process found in graphene in sec. VI.2.2 produces the gap in TaS_2. The gap $\Delta = (16.0 \pm 4.5)\,\mathrm{meV}$ could be explained with an inelastic tunneling process involving the amplitude mode near the Γ point with an energy around the calculated $E_0 = 10.6\,\mathrm{meV}$. Confirmation of this explanation would need an ab initio model of the electron-phonon interaction and its effects on the tunneling, similarly to graphene [38].

VII Conclusion

The benchmarks on QUANTUM ESPRESSO's `PWscf` and `PHonon` modules carried out in this thesis show how system size, compilers and parallelization parameters interplay to influence scaling of calculation across multiple processors. TaS_2, the system with longer times per iteration has a different scaling behavior to silicon, the system with shorter times per iteration: the former shows linear speedup for significantly more processors than the latter. Using the Intel oneAPI instead of the OpenMPI module leads to different scaling behavior on silicon, while minimal runtime stays the same. This is due to the slightly longer single core performance with Intel oneAPI carrying over into calculation of the speedup. In calculations on TaS_2, using Intel oneAPI compilers leads to the calculations scaling well not just on a single node, but on almost two full nodes.

The different ranges for the optimal number of processors carries over to the choice of parameters for QUANTUM ESPRESSO's k-point parallelization. For calculations on both systems, the optimal size of processor pools for k-point parallelization is near the number of processors where the speedup does not follow ideal scaling anymore. For the system sizes at hand, utilizing linear-algebra parallelization has no impact in most cases and slows down the calculation in the case of electronic structure calculations on silicon. In phonon calculations, using a combination of k-point and image parallelization leads to the calculations scaling over a wide range of processors.

The calculated phonon modes and frequencies on TaS_2 in the charge-density-wave phase give a possible explanation for a gap around the Fermi level observed in an STS experiment on the material. This gap forms via an inelastic tunneling process involving the amplitude mode near the Γ point opening up just for electron energies higher than that of this phonon mode. While the energy does not fit perfectly to the size of the gap, the symmetric form of the gap would be explained. Further theoretical work is needed to confirm this proposition.

Bibliography

[1] J. Hall et al. "Environmental Control of Charge Density Wave Order in Monolayer 2H-TaS2". In: *ACS Nano* 13.9 (Sept. 24, 2019). Publisher: American Chemical Society, pp. 10210–10220. ISSN: 1936-0851. DOI: 10.1021/acsnano.9b03419.

[2] P. Hohenberg and W. Kohn. "Inhomogeneous Electron Gas". In: *Phys. Rev.* 136.3 (Nov. 1964). Publisher: American Physical Society, B864–B871. DOI: 10.1103/PhysRev.136.B864.

[3] W. Kohn and L. J. Sham. "Self-Consistent Equations Including Exchange and Correlation Effects". In: *Phys. Rev.* 140.4 (Nov. 1965). Publisher: American Physical Society, A1133–A1138. DOI: 10.1103/PhysRev.140.A1133.

[4] R. O. Jones. "Density functional theory: Its origins, rise to prominence, and future". In: *Rev. Mod. Phys.* 87.3 (Aug. 2015). Publisher: American Physical Society, pp. 897–923. DOI: 10.1103/RevModPhys.87.897.

[5] P. Giannozzi et al. "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials". In: *Journal of Physics: Condensed Matter* 21.39 (Sept. 2009). Publisher: IOP Publishing, p. 395502. DOI: 10.1088/0953-8984/21/39/395502.

[6] P. Giannozzi et al. "Advanced capabilities for materials modelling with Quantum ESPRESSO". In: *Journal of Physics: Condensed Matter* 29.46 (Oct. 2017). Publisher: IOP Publishing, p. 465901. DOI: 10.1088/1361-648x/aa8f79.

[7] R. G. Dickinson and L. Pauling. "THE CRYSTAL STRUCTURE OF MOLYBDENITE". In: *Journal of the American Chemical Society* 45.6 (June 1, 1923). Publisher: American Chemical Society, pp. 1466–1471. ISSN: 0002-7863. DOI: 10.1021/ja01659a020.

[8] K. S. Novoselov et al. "Two-dimensional atomic crystals". In: *Proceedings of the National Academy of Sciences* 102.30 (2005). _eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.0502848102, pp. 10451–10453. DOI: 10.1073/pnas.0502848102.

[9] D. N. Basov, R. D. Averitt, and D. Hsieh. "Towards properties on demand in quantum materials". In: *Nature Materials* 16.11 (Nov. 1, 2017), pp. 1077–1088. ISSN: 1476-4660. DOI: 10.1038/nmat5017.

[10] M. Born and R. Oppenheimer. "Zur Quantentheorie der Molekeln". In: *Annalen der Physik* 389.20 (Jan. 1, 1927). Publisher: John Wiley & Sons, Ltd, pp. 457–484. ISSN: 0003-3804. DOI: 10.1002/andp.19273892002.

[11] N. Marzari. "Ab-initio Molecular Dynamics for Metallic Systems". PhD thesis. University of Cambridge, 1996.

[12] R. M. Martin. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511805769.

[13] J. Harris. "Adiabatic-connection approach to Kohn-Sham theory". In: *Phys. Rev. A* 29.4 (Apr. 1984). Publisher: American Physical Society, pp. 1648–1659. DOI: 10.1103/PhysRevA.29.1648.

[14] J. P. Perdew, K. Burke, and M. Ernzerhof. "Generalized Gradient Approximation Made Simple". In: *Phys. Rev. Lett.* 77.18 (Oct. 1996). Publisher: American Physical Society, pp. 3865–3868. DOI: 10.1103/PhysRevLett.77.3865.

[15] D. R. Hamann, M. Schlüter, and C. Chiang. "Norm-Conserving Pseudopotentials". In: *Phys. Rev. Lett.* 43.20 (Nov. 1979). Publisher: American Physical Society, pp. 1494–1497. DOI: 10.1103/PhysRevLett.43.1494.

[16] S. Baroni et al. "Phonons and related crystal properties from density-functional perturbation theory". In: *Rev. Mod. Phys.* 73.2 (July 2001). Publisher: American Physical Society, pp. 515–562. DOI: 10.1103/RevModPhys.73.515.

[17] R. P. Feynman. "Forces in Molecules". In: *Phys. Rev.* 56.4 (Aug. 1939). Publisher: American Physical Society, pp. 340–343. DOI: 10.1103/PhysRev.56.340.

[18] P. D. DeCicco and F. A. Johnson. "The Quantum Theory of Lattice Dynamics. IV". In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 310.1500 (1969). Publisher: The Royal Society, pp. 111–119. ISSN: 00804630. URL: http://www.jstor.org/stable/2416303 (visited on 06/24/2022).

[19] R. M. Pick, M. H. Cohen, and R. M. Martin. "Microscopic Theory of Force Constants in the Adiabatic Approximation". In: *Phys. Rev. B* 1.2 (Jan. 1970). Publisher: American Physical Society, pp. 910–920. DOI: 10.1103/PhysRevB.1.910.

[20] S. Baroni, P. Giannozzi, and A. Testa. "Green's-function approach to linear response in solids". In: *Phys. Rev. Lett.* 58.18 (May 1987). Publisher: American Physical Society, pp. 1861–1864. DOI: 10.1103/PhysRevLett.58.1861.

[21] X. Gonze. "Adiabatic density-functional perturbation theory". In: *Phys. Rev. A* 52.2 (Aug. 1995). Publisher: American Physical Society, pp. 1096–1114. DOI: 10.1103/PhysRevA.52.1096.

[22] G. Hager and G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. 0th ed. CRC Press, July 2, 2010. ISBN: 978-1-4398-1193-1. DOI: 10.1201/EBK1439811924.

[23] G. M. Amdahl. "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities". In: *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*. AFIPS '67 (Spring). event-place: Atlantic City, New Jersey. New York, NY, USA: Association for Computing Machinery, 1967, pp. 483–485. ISBN: 978-1-4503-7895-6. DOI: 10.1145/1465482.1465560.

[24] *Quantum ESPRESSO User's Guide (v. 7.0)*. URL: https://www.quantum-espresso.org/documentation/ (visited on 05/23/2022).

[25] *PWscf User's Guide (v. 7.0)*. URL: https://www.quantum-espresso.org/documentation/package-specific-documentation/ (visited on 05/23/2022).

[26] *PHonon User's Guide (v. 7.0)*. URL: https://www.quantum-espresso.org/documentation/package-specific-documentation/ (visited on 05/23/2022).

[27] Y. J. Chabal, ed. *Fundamental Aspects of Silicon Oxidation*. Springer Berlin Heidelberg, 2001. DOI: 10.1007/978-3-642-56711-7.

[28] D. R. Hamann. "Erratum: Optimized norm-conserving Vanderbilt pseudopotentials [Phys. Rev. B 88, 085117 (2013)]". In: *Phys. Rev. B* 95.23 (June 2017). Publisher: American Physical Society, p. 239906. DOI: 10.1103/PhysRevB.95.239906.

[29] A. Kokalj. "XCrySDen—a new program for displaying crystalline structures and electron densities". In: *Journal of Molecular Graphics and Modelling* 17.3 (1999). Code available

from http://www.xcrysden.org/., pp. 176–179. ISSN: 1093-3263. DOI: https://doi.org/10.1016/S1093-3263(99)00028-5.

[30] J. A. Wilson and A. D. Yoffe. "The transition metal dichalcogenides discussion and interpretation of the observed optical, electrical and structural properties". In: *Advances in Physics* 18.73 (1969). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00018736900101307, pp. 193–335. DOI: 10.1080/00018736900101307.

[31] S. Nagata et al. "Superconductivity in the layered compound 2H-TaS2". In: *Journal of Physics and Chemistry of Solids* 53.10 (1992), pp. 1259–1263. ISSN: 0022-3697. DOI: https://doi.org/10.1016/0022-3697(92)90242-6.

[32] J. A. Wilson, F. J. Di Salvo, and S. Mahajan. "Charge-Density Waves in Metallic, Layered, Transition-Metal Dichalcogenides". In: *Phys. Rev. Lett.* 32.16 (Apr. 1974). Publisher: American Physical Society, pp. 882–885. DOI: 10.1103/PhysRevLett.32.882.

[33] E. Navarro-Moratalla et al. "Enhanced superconductivity in atomically thin TaS2". In: *Nature Communications* 7.1 (Apr. 2016), p. 11043. ISSN: 2041-1723. DOI: 10.1038/ncomms11043.

[34] G. Grüner. "The dynamics of charge-density waves". In: *Rev. Mod. Phys.* 60.4 (Oct. 1988). Publisher: American Physical Society, pp. 1129–1181. DOI: 10.1103/RevModPhys.60.1129.

[35] C. Hartwigsen, S. Goedecker, and J. Hutter. "Relativistic separable dual-space Gaussian pseudopotentials from H to Rn". In: *Phys. Rev. B* 58.7 (Aug. 1998). Publisher: American Physical Society, pp. 3641–3662. DOI: 10.1103/PhysRevB.58.3641.

[36] K. Oura et al. *Surface Science*. Springer Berlin Heidelberg, 2003. DOI: 10.1007/978-3-662-05179-5.

[37] Y. Zhang et al. "Giant phonon-induced conductance in scanning tunnelling spectroscopy of gate-tunable graphene". In: *Nature Physics* 4.8 (Aug. 1, 2008), pp. 627–630. ISSN: 1745-2481. DOI: 10.1038/nphys1022.

[38] T. O. Wehling et al. "Phonon-Mediated Tunneling into Graphene". In: *Phys. Rev. Lett.* 101.21 (Nov. 2008). Publisher: American Physical Society, p. 216803. DOI: 10.1103/PhysRevLett.101.216803.

[39] L. Vitali et al. "Phonon and plasmon excitation in inelastic electron tunneling spectroscopy of graphite". In: *Phys. Rev. B* 69.12 (Mar. 2004). Publisher: American Physical Society, p. 121414. DOI: 10.1103/PhysRevB.69.121414.

Acknowledgement

I thank my supervisor Professor Tim Wehling for allowing me to get a glimpse into the fascinating research he and his research group do and for the opportunity to learn a lot about condensed matter in the past 5 months.

I also thank Michael Winter and Jan Berges for direct support, scripts, data, LaTeX templates and much more as well as the whole group "Computational Condensed Matter Theory" for welcoming me into their open and productive atmosphere.

I thank all developers of free software, especially the ones I used in this thesis: Quantum ESPRESSO, LaTeX, Python, Numpy, Matplotlib, Git.

I thank my family for support through my all my studies up until now.

I especially thank Liv for massive amounts of love and support during the writing of this thesis.

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Eidesstattliche Erklärung

Ich versichere, dass ich die beigefügte schriftliche Bachelorarbeit selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem einzelnen Fall unter genauer Angabe der Quelle deutlich als Entlehnung kenntlich gemacht. Dies gilt auch für alle Informationen, die dem Internet oder anderer elektronischer Datensammlungen entnommen wurden.
Ich erkläre ferner, dass die von mir angefertigte Bachelorarbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung im Rahmen meines Studiums war. Die von mir eingereichte schriftliche Fassung entspricht jener auf dem elektronischen Speichermedium.
Ich bin damit einverstanden, dass die Bachelorarbeit veröffentlicht wird

Ort, Datum	Unterschrift