Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelorthesis

Title

vorgelegt von

TJARK SIEVERS

Fakultät: Mathematik, Informatik und Naturwissenschaften
Fachbereich: Physik
Studiengang: Physik
Matrikelnummer: 7147558
Erstgutachter:
Zweitgutachter:

Kurzzusammenfassung

Contents

Kurzzusammenfassung **i**

Danksagung **ix**

Motivation **xi**

I Many-body physics **1**
- I.1 The electronic structure problem . . . 1
- I.2 Density Functional Theory . . . 1
  - I.2.1 Hohenberg-Kohn theorems . . . 2
  - I.2.2 Kohn-Sham equations . . . 3
  - I.2.3 Choice of basis set and pseudopotentials . . . 3
- I.3 Density Functional Perturbation Theory . . . 3

II Computational Basics **5**
- II.1 Parallel computing . . . 5
  - II.1.1 On scalabilty . . . 5
- II.2 QUANTUM ESPRESSO . . . 6

III Parallelisation of self-consistent calculations of electronic-structure properties **7**
- III.1 First scaling tests . . . 7
- III.2 Testing different compilers and mathematical libraries . . . 8
- III.3 Using the parallelisation parameters of QUANTUM ESPRESSO . . . 9
  - III.3.1 Comparison with calculations on the HLRN cluster . . . 11
  - III.3.2 Conclusion: Parameters for optimal scaling . . . 11

IV Parallelization of DFPT calculations **13**

Literaturverzeichnis **15**

Listings **17**
- List of Figures . . . 17
- List of Tables . . . 17

Glossary

Intel OneAPI . 9, 10

OpenBLAS Open-source implementation of the BLAS (Basic Linear Algebra Subprograms) and LAPACK APIs. 8, 9

OpenMPI Open MPI is a Message Passing Interface (MPI) library project. 7–10, 17

Scalapack . 9, 10

Acronyms

DFT Density Functional Theory. 2

Danksagung

Motivation

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 Many-body physics

I.1 The electronic structure problem

In solid state physics, the Hamiltonian describing the interacting nuclei and electrons in the solid is well known, as all interactions except Coulomb interaction can safely be ignored at the mass and energy scales at which the electrons and nuclei reside. This very general problem consisting of both the electronic and nuclei degrees of freedom can be simplified in a first step by employing the Born-Oppenheimer approximation. 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 $\mathbf{R}_\alpha$ as a parameter.

This problem is described by the time-independent Schrödinger equation

$$\hat{H}\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N) = E\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N) \tag{I.1}$$

with the Hamiltonian in first quantization (i running over the electrons, α, β over the nuclei)

$$\begin{aligned}\hat{H} &= \hat{T}_e + \hat{U}_{e-e} + \hat{V}_{n-e} + \hat{W}_{n-n} && \text{(I.2)}\\ &= -\sum_i \frac{1}{2}\nabla_i^2 + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} - \sum_i\sum_\alpha \frac{Z_\alpha}{|\mathbf{r}_i - \mathbf{R}_\alpha|} + \frac{1}{2}\sum_{\alpha\beta}\frac{Z_\alpha Z_\beta}{R_{\alpha\beta}} && \text{(I.3)}\end{aligned}$$

where:

- $\hat{T}_e$ is the kinetic energy of the electrons
- $\hat{U}_{e-e}$ is the Coulomb interaction between the electrons and
- $\hat{V}_{n-e}$ is the potential energy of the electrons in the field of the nuclei
- $\hat{W}_{n-n}$ is the Coulomb interaction between the nuclei

The terms $\hat{V}_{n-e}$ and $\hat{W}_{n-n}$ can then be combined into an external potential V for the interacting electrons, so that the Hamiltonian reads

$$\hat{H} = \hat{T} + \hat{U} + \hat{V} \tag{I.4}$$

This Hamiltonian will be used in the further development of the underlying theory for this thesis.

I.2 Density Functional Theory

A direct solution to the electronic structure problem, this meaning obtaining the ground-state many-body wave function $\Psi(\mathbf{r_1},\ldots,\mathbf{r}_N)$ for a given potential is analytically impossible even for a small number of electrons compared to the number of electrons in a macroscopic crystal. 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 developed, the outline of which can be found in any literature on solid state physics [1].

I.2.1 Hohenberg-Kohn theorems

The starting for DFT is the exact reformulation of the outlined electronic structure problem by Hohenberg and Kohn [2]. This reformulation uses the ground state density of the electronic system $n_o(r)$ as the basic variable. To achieve this, Hohenberg and Kohn [2] formulated two theorems, 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 paper [2] or any publication on DFT [1]):

I The external potential (and via the Schrödinger equation also the ground state wave function and the ground state energy) is a unique functional of the ground state density (except for an additive constant).

II The ground state energy minimizes the energy functional,

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

.

The proof of those theorems show the existence of the energy functional $E[n(\mathbf{r})]$, 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

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

Defining the universal functional $F[n(\mathbf{r})] = \langle \Psi|T+U|\Psi\rangle$, which is material independent and writing $n(\mathbf{r}') = \Psi^*(\mathbf{r}')\Psi(\mathbf{r}')$, the energy functional becomes

$$E[n(\mathbf{r})] = F[n(\mathbf{r})] + \int \mathrm{d}\mathbf{r}'\, V(\mathbf{r}')n(\mathbf{r}') \tag{I.5}$$

This is just a formal definition, as all the formerly mentioned complication of the Hamiltonian I.4 now lie in the functional $F[n(\mathbf{r})]$. With a known or well approximated universal functional $F[n(\mathbf{r})]$, 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 3 spatial coordinates instead of $3N$ coordinates with the full Hamiltonian.

I.2.2 Kohn-Sham equations

One way of approximating the functional $F[n]$ was given by Kohn and Sham [3]. The idea is to use a non-interacting auxiliary system of electrons

$$H_0 = \sum_{i}^{N_e} \frac{p_i^2}{2m} + v_{KS}(\mathbf{r}_i) \tag{I.6}$$

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 wave functions, the solutions to the non-interacting problem Ψ_i. This gives rise to a kinetic energy

this is shit

I.2.3 Choice of basis set and pseudopotentials

this whole thing

I.3 Density Functional Perturbation Theory

hate thinking about it

II Computational Basics

II.1 Parallel computing

The following section will give an overview of the technical aspects of running computer code (such as QUANTUM ESPRESSO) on massively parallel computing environments (such as the PHYSnet compute cluster). The information presented can be found in any textbook on parallel or high-perfomance computing [4].

II.1.1 On scalabilty

In scientific computing, one can identify two distinct reasons to distribute workloads 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 grow outside the capabilities of a single core

In order to judge how well a task can parallelized, usually some sort of scalabilty 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 much of the resources is used for solving the problem?

The speedup by using N workers to solve a problem instead of one is defined as $S = \frac{T_1}{T_N}$, 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, all processors need the same time for their respective workloads and don't have to wait for others processors to finish their workload to continue, the execution time on N processors would be $\frac{T_1}{N}$, so the speedup would be $S = \frac{T_1}{\frac{T_1}{N}} = N$.

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 programm 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, faster parallel code execution can come from:

- *Better caching*: when the data the program is working with is distributed among processors (assuming constant problem size), it may enable the data to be stored in faster cache memory. Modern computers typically have three layers of cache memory, with level 1 cache being the smallest and fastest and level 3 being the largest and slowest, so smaller data chunks per processor can lead to the data not being stored in main memory, but completely in cache or in a faster cache level

II.2 Quantum ESPRESSO

III Parallelisation of self-consistent calculations of electronic-structure properties

III.1 First scaling tests

The first step in analysing the scaling of Quantum ESPRESSO is to perform a baseline scaling test without any optimisations appplied. In Fig. III.1 and III.2 two scaling tests on the earlier mentioned benchmarking systems Si and TaS2 are pictured. The tests are run using Quantum ESPRESSO 7.0, compiled using the Fortran and C compilers in OpenMPI 4.1.0, without any of compilation or runtime optimisation parameters mentioned in section II.2 used.

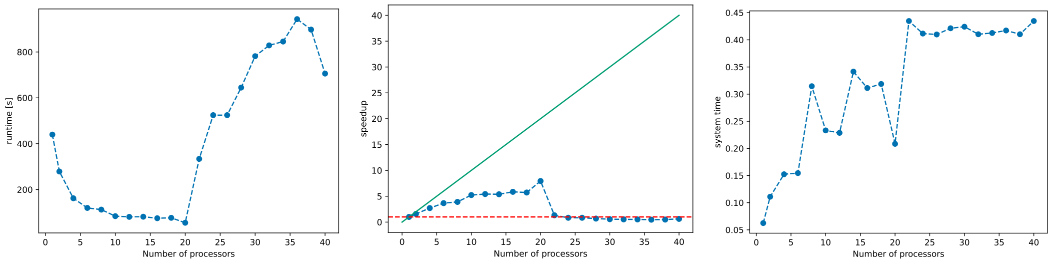


Figure III.1: *Baseline scaling test on the Si benchmarking system* Quantum ESPRESSO 7.0, OpenMPI 4.1.0, `nk 1` and `nd 1`

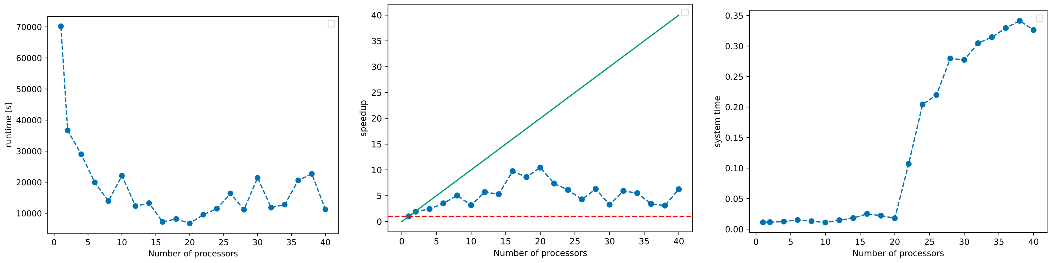


Figure III.2: *Baseline scaling test on the TaS2 benchmarking system* Quantum ESPRESSO 7.0, OpenMPI 4.1.0, `nk 1` and `nd 1`

Three different metrics of scalability are pictured in **??**.

- runtime: absolute runtime (walltime) of the compute job

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

For further analysis mainly speedup will be used as a metric of scalability, because it lends itself to easy interpretation: optimal scalability is achieved when the speedup scales linearly with the number of processors (with a slope of one), as discussed in ch. II.1. This on the one hand necessarily implies good parallelization and a lower runtime for more processors used, but the other two parameters should also always be considered.

As an example, for 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 needing to wait a whole day for the calculation and being able to let the job run overnight and continue working in the morning, so using 100 more processors for that can be considered a good use of resources.

On a single node, both the Si and TaS2 calculations show good, but not perfect scaling behavior: the speedup does approximately scale linearly with the number of processors, but the slope is closer to $\frac{1}{2}$, than 1. Even though the scaling behavior is not perfect, there is just a small, almost constant amount of runtime used by system calls, this speaks for good parallelization. As discussed in sec. II.1, startup time is an unavoidable part of every parallel program, so a constant amount of time used not for calculations is expected, bad parallelization on the other hand shows itself by introducing waiting times between processors, which makes the waiting time in some way dependent on the number of processors.

When using more than one node, not only does the scaling get worse, the execution needs longer than on a single core for the Si system, with a marginally better performance for the TaS2 system. This is also seen in the plots of system time. The percentage of time used for tasks not directly related to calculations goes from a near constant value for under 20 processors to 50% of the execution time for the Si system and 35% for the TaS2 system.

These scaling tests pose now two questions to be answered:

- Is better scaling on a single node possible?
- How can acceptable scaling over more than one node be achieved?

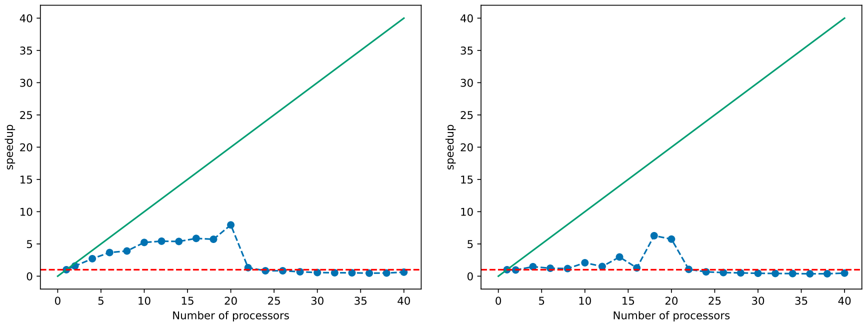
III.2 Testing different compilers and mathematical libraries

A first strategy for solving issues with parallelization is trying different compilers and mathematical libraries. In the PHYSnet cluster a variety of software packages is available. As discussed in sec. II.2, for the compilation of Quantum ESPRESSO

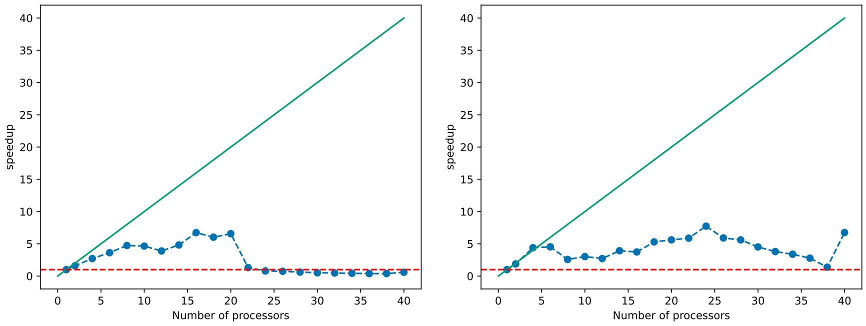
For testing Quantum ESPRESSO will be compiled using the following software combinations:

- OpenMPI 4.1.0 and OpenBLAS

- OpenMPI 4.1.0 and Scalapack
- Intel OneAPI 2021.4 (includes Intel MPI, Fortran and C compilers as well as Intel MKL, a scalable mathematical library)



(a) *OpenMPI* **(b)** *OpenMPI + OpenBLAS*



(c) *OpenMPI + Scalapack* **(d)** *Intel OneAPI*

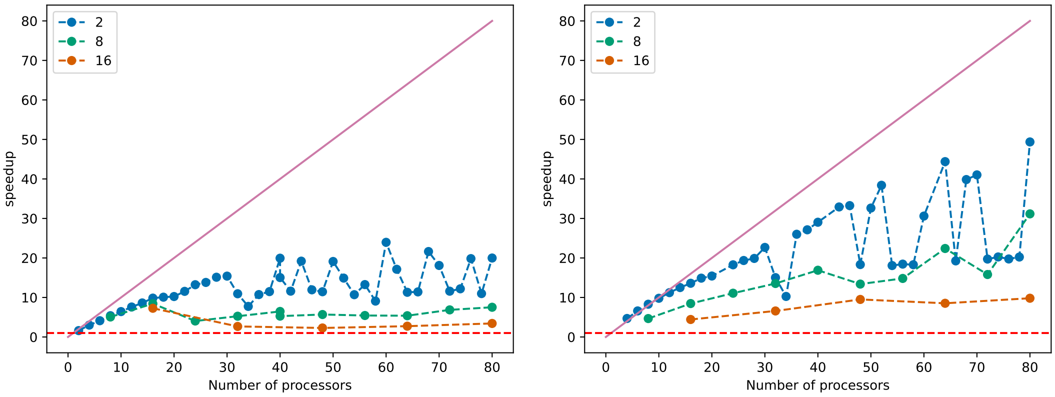
Figure III.3: *Baseline scaling test with different combinations of compilers and mathematical libraries*

analysis

III.3 Using the parallelisation parameters of Quantum ESPRESSO

As detailed in section II.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.

The benchmark pictured in III.4 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. III.4 shows the scaling for poolsizes 2, 8 and 16 for Quantum ESPRESSO being compiled with OpenMPI/Scalapack and Intel oneAPI. This choice of pool sizes showcases the smallest pool size possibly (namely 2), as well as a bigger pool size with 16, that still gives rise to a few data points over the chosen range of processors.



(a) *OpenMPI + Scalapack*

(b) *Intel OneAPI*

Figure III.4: *Benchmark with k-point parallelization for the Si benchmarking system*

analysis

Fig. III.4 shows the significant difference between using

The same scaling test is applied to the TaS2 system in fig. III.5, with the same list of pool sizes, but over a wider range of processors.

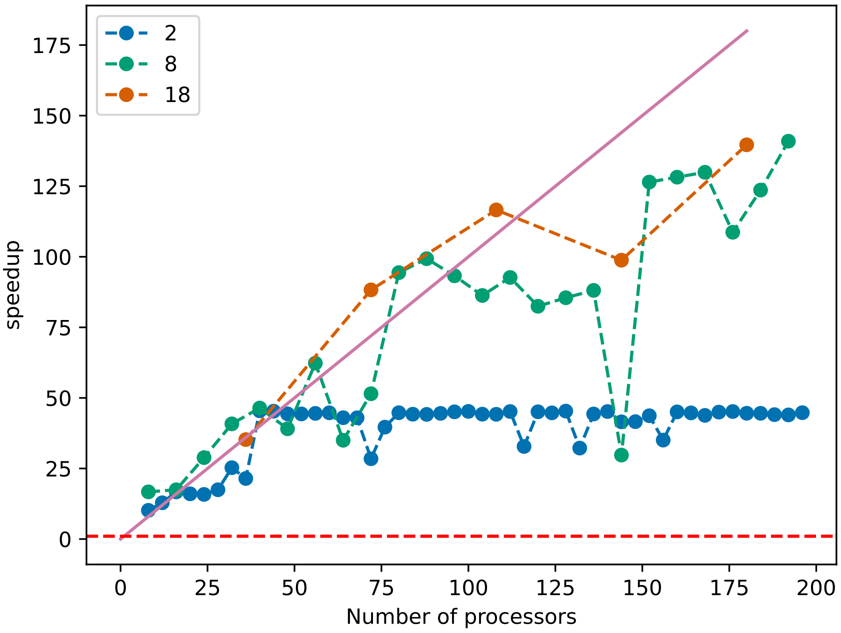


Figure III.5: *Benchmark with k-point parallelization for the TaS2 benchmarking system*

Remarkably, the scaling behavior is swapped in comparison to III.4, as the pool size 2 saturates fast and the bigger pool sizes show way better scaling behavior. Furthermore, there are instances of better than linear scaling, which according to QUANTUM ESPRESSO docs can be attributed to better caching of data.

It can also be instructive to look at the idle time for this benchmark to judge the quality of parallelization.

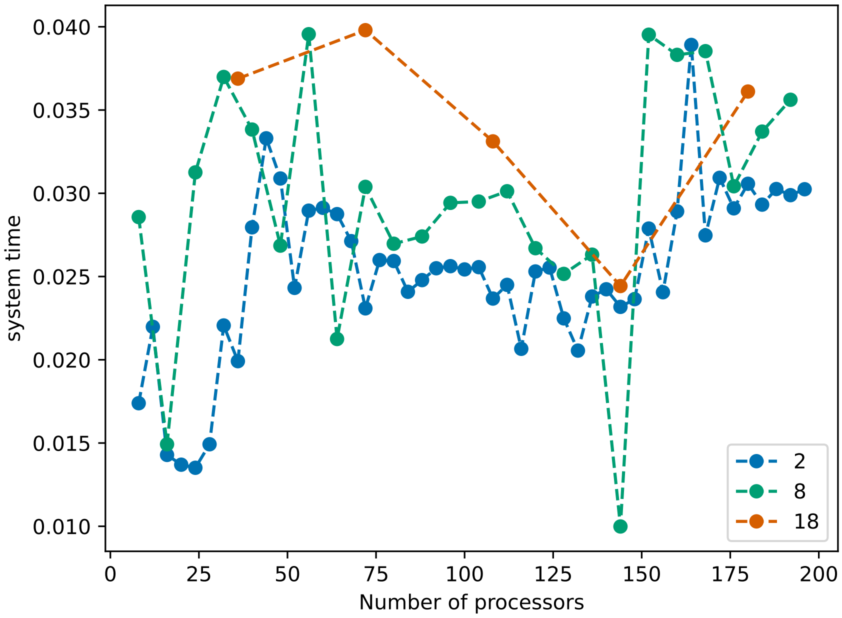


Figure III.6: *XXX*

Fig. III.6 shows a distribution of idle times between 1 % and 4 % of the whole wall time, without any kind of systemic increase over any range of processors. This means the parallelization is as good as possible for these

III.3.1 Comparison with calculations on the HLRN cluster

III.3.2 Conclusion: Parameters for optimal scaling

IV Parallelization of DFPT calculations

Bibliography

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

[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] 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`.

Listings

List of Figures

III.1 Baseline scaling test on the Si benchmarking system *Quantum ESPRESSO 7.0, OpenMPI 4.1.0, `nk 1` and `nd 1`* . . . 7
III.2 Baseline scaling test on the TaS2 benchmarking system *Quantum ESPRESSO 7.0, OpenMPI 4.1.0, `nk 1` and `nd 1`* . . . 7
III.3 Baseline scaling test with different combinations of compilers and mathematical libraries . . . 9
III.4 Benchmark with k-point parallelization for the Si benchmarking system . . . 10
III.5 Benchmark with k-point parallelization for the TaS2 benchmarking system . . . 10
III.6 XXX . . . 11

List of Tables