# Evaluating Eigensolver Schemes within the Density Functional Theory Package WIEN2k

Thomas Ruh and Peter Blaha
Institute of Materials Chemistry, TU Wien,
Getreidemarkt 9/165-TC, A-1060 Vienna, Austria

**POSTER PAPER**

*Abstract*—**WIEN2k is a program package that utilizes Density Functional Theory (DFT) to describe materials and calculate their properties. This is done by numerically solving the Schrödinger equation. The most expensive part of such calculations in terms of consumption of computational resources is the numerical solution of hermitian (or symmetric) generalized eigenvalue problems that involve large matrices up to dimensions of 100.000 – depending on the complexity of the material. Therefore, efficient use of HPC systems is critical to be able to run such calculations in a reasonable time.**
**In this work the present performance of parallel eigensolver schemes already implemented in WIEN2k is evaluated in order to find bottlenecks.**

*Keywords:* Scalable Computing, Benchmarking, Eigensolvers, Density Functional Theory, Electronic Structure, WIEN2k

## I. Introduction

Nowadays, advanced materials become more and more complex, because specific applications require corresponding materials with desired properties. An understanding of the relation between those interesting properties and certain characteristics (e.g. structure or composition) is crucial in order to predict properties or design materials [1]. A possible way to gain such an understanding is through simulations. There exist many different methods, ranging from empirical methods (using material specific parametrized forcefields to simulate for instance proteins [2], [3]) and Monte Carlo methods (that use repeated random sampling) [4], [5] to so-called first-principle (or ab-initio) methods such as Density Functional Theory (DFT) [1], [6], [7].

The program package WIEN2k [8] uses DFT to simulate a wide range of materials from first principles, i.e. no material specific input parameters are needed. The only input that is necessary is (a rough guess of) the structure of the given material, which can be anything from solids (both infinite "perfect" bulk solids and solids with vacancies, impurities, etc.), nanostructures [9] and surfaces [10] to molecules.
The properties that can be obtained from those calculations are numerous and include, for example:

- Structural properties: geometry, exact atomic positions – for instance for hydrogen in solids, which cannot be easily accessed experimentally, phase transitions.
- Electronic properties: band structure, density of states, spectroscopies (IR, Raman, XPS, XAS).
- Optical properties: absorbance, reflectance.
- Other properties: magnetism, NMR-Knight-Shifts, Scanning Tunneling Microscopy (STM) images of surfaces.

The computational costs of such calculations depend strongly on the complexity of the material and on the accuracy requirements for a given property. The size of the corresponding eigenvalue problem (i.e. the dimension of the involved matrices) grows the more complex a given system gets, since one needs about 100 to 500 basis functions per atom in the unit cell. "Simple" structures with about 50 atoms in the unit cell and structures with inversion symmetry can easily be simulated on a modern PC within a reasonable amount of time. However, since the computational effort scales with $n^3$ (where $n$ is the number of atoms per unit cell) more complex materials or larger unit cells are much more expensive, both in terms of computation time and memory consumption. For instance, simulations of surfaces to explain complicated surface reconstructions [10] require large supercells with several hundreds of atoms. Another example for rather costly calculations are NMR-Knight-shifts of metals, since the Schrödinger equation must be solved for millions of k-points [11]. Thus it is vital to utilize HPC systems that allow the simultaneous usage of large numbers of cores. The main programs of WIEN2k are already parallelized and have different eigensolver schemes implemented. The purpose of the present paper is to test the scaling behavior of these eigensolvers and identifying possible bottlenecks and limitations when using WIEN2k in a high performance computing context.

## II. Density Functional Theory

The time independent Schrödinger equation (TISE) can be used to describe a given stationary system (an atom, a molecule, or a solid):

$$\hat{H}\Psi(\mathbf{r_1}, \mathbf{r_2}, ..., \mathbf{r_n}) = E\Psi(\mathbf{r_1}, \mathbf{r_2}, ..., \mathbf{r_n}), \qquad (1)$$

where $\hat{H}$ and $\Psi$ are the Hamiltonian and the wave function, respectively, $\mathbf{r}$ are the coordinates of the $n$ electrons, and $E$ is the energy. The Hamiltonian has the general form:

$$\hat{H} = -\frac{1}{2}\sum_i \nabla_i^2 + v_{\text{ext}}(\mathbf{r}) + \frac{1}{2}\sum_{j\neq i}\sum_i \frac{1}{|\mathbf{r_j} - \mathbf{r_i}|}, \qquad (2)$$

where $\nabla^2$ corresponds to the kinetic energy of electron $i$, $v_{\text{ext}}$ is an external potential acting on the electrons (i.e. produced by the nuclear charges) and the third term describes the repulsive electron-electron interaction. $\hat{H}$ does not contain terms depending on the motion of the nuclei. This motion can be neglected in accordance with the Born-Oppenheimer approximation [12] – the nuclear positions enter into (2) as parameters (in the form of a nuclear potential acting on the electrons).
One can, in principal, learn all information about the system by finding the wave function $\Psi$. However, the TISE cannot be solved analytically for systems with more than one electron, and even approximate numerical methods are limited to systems with a few electrons, because of the multi-dimensionality of $\Psi(\mathbf{r_1}, \mathbf{r_2}, ..., \mathbf{r_n})$. A more practical way to solve such problems is facilitated by Density Functional Theory (DFT).

The basic principle of DFT is the Hohenberg-Kohn theorem [6], which expresses the energy of an $n$-electron system as a functional $F$ of the electron density $\rho$:

$$E = \int v_{\text{ext}}(\mathbf{r})\rho(\mathbf{r})d\mathbf{r} + F[\rho]. \qquad (3)$$

This energy is still exact, but only dependent on 3 (instead of $3n$) variables, thus reducing computational costs significantly. The problem, however, is that the functional $F$ is not known. Therefore, this principle functional is decomposed in the (still exact) Kohn-Sham-DFT formalism [7], leading to a system of $n$ single-electron wavefunctions $\Psi_i$ for non-interacting electrons, that yield the exact electron density:

$$\left\{-\frac{1}{2}\nabla^2 + v_{\text{ext}} + v_C(\rho(\mathbf{r})) + v_{xc}(\rho(\mathbf{r}))\right\}\Psi_i(\mathbf{r}) = \epsilon_i\Psi_i(\mathbf{r}), \qquad (4)$$

where $\nabla^2$ corresponds to the non-interacting kinetic energy, $v_{\text{ext}}$ describes the nuclear potential, that acts on the electron, $v_C$ is the repulsive Coloumb- or Hartree potential due to interaction with the other electrons (only dependent on the total electron density), and $v_{xc}$ represents the exchange-correlation potential of the system of interacting particles with the same $\rho(\mathbf{r})$. Only the last term is unknown, and must be approximated, but there exist many well-defined functionals that can be used to approximate this potential [13].

Since the Kohn-Sham equation (4) cannot be solved analytically, the Kohn-Sham wavefunctions $\Psi_i$ are expanded into a set of (non-orthogonal) basis functions $\phi_n$:

$$\Psi(\mathbf{r}) = \sum_n c_n\phi_n(\mathbf{r}), \qquad (5)$$

where $c_n$ are the coefficients of the linear combination.

For a WIEN2k calculation the unit cell is divided into two different regions with different basis-sets [8]:

- Non-touching atomic spheres at the positions of the nuclei, where the basis functions consist of a radial and an angular-momentum dependent part. In (6), $A_{lm}$ are coefficients (for the quantum numbers $l$ and $m$) used to match the solution inside the sphere to the one outside, $u_l(r)$ are solutions of the radial Schrödinger equation, and $Y_{lm}(\mathbf{r})$ are so-called spherical harmonics that are defined for sets of $l$ and $m$.

$$\phi_n = \begin{cases} \sum_{lm} A_{lm}(\mathbf{K_n})u_l(r)Y_{lm}(\mathbf{r}), & \text{inside the spheres} \\ e^{i\mathbf{K_n}\mathbf{r}}, & \text{outside the spheres (interstitial)} \end{cases} \qquad (6)$$

- The interstitial region between the atoms, where the wavefunctions are expanded into plane waves where $\mathbf{K_n}$) denotes a full reciprocal lattice vector.

The coefficients $c_n$ are determined using the variational principle:

$$\frac{\partial E}{\partial c_n} = 0, \qquad (7)$$

which leads to a set of $n$ equations, that can be rewritten as a generalized eigenvalue problem of the form:

$$HC = ESC, \qquad (8)$$

where $H$ and $S$ are the Hamilton and the overlap matrix, respectively, the matrix $C$ contains the eigenvectors, and $E$ is a diagonal matrix containing the eigenvalues.

The matrix elements of the matrices $H$ and $S$ can be derived from:

$$H_{m,n} = \int \phi_m^*\hat{H}\phi_n d\mathbf{r}, \qquad (9)$$

$$S_{m,n} = \int \phi_m^*\phi_n d\mathbf{r}. \qquad (10)$$

In order to setup $H$ and solve for $\Psi_i$ one needs an electron density $\rho(\mathbf{r})$. However, $\rho(\mathbf{r})$ can only be calculated from the $\Psi_i$:

$$\rho(\mathbf{r}) = \sum_{\text{occupied}} |\Psi_i|^2. \qquad (11)$$

Therefore an iterative procedure is necessary: Starting from an estimate for $\rho$, spherical and non-spherical potentials $v_{\text{sp}}$ and $v_{\text{nsp}}$ are calculated. Those potentials are in turn used to recalculate basis functions $\phi_n$, solve the eigenvalue problem (8) and find the Kohn-Sham orbitals $\Psi_i$ and their energies $\epsilon_i$. A “new” electron density is then generated from those orbitals. This procedure is repeated in the scf cycle until certain convergence criteria are met, i.e. some properties (for instance the energy) are the same before and after a step within a given accuracy.

## III. Eigensolver schemes in WIEN2k

The most expensive steps in terms of computation time using WIEN2k are the setup and solution of a symmetric (when inversion symmetry is present) or hermitian generalized eigenvalue problem. Those two steps, that lead to the calculation of the eigenvalues $\epsilon_i$ and the corresponding eigenvectors $\Psi_i$, are combined within one program (called `lapw1`), and take up more than 90% of the total time per cycle.

### A. Setup of matrices $H$ and $S$

The setup of the Hamilton matrix elements $H$ and the overlap matrix $S$, respectively, is done according to (9) and (10) in two steps:

- Setup of spherical matrix elements $H_{\text{sp}}$ and $S_{\text{sp}}$.
- Setup of non-spherical matrix elements $H_{\text{nsp}}$ and $S_{\text{nsp}}$.

The corresponding subprograms have been parallelized "by hand" using MPI and rely mainly on (local) BLAS [14] routines (the level 3 routine `xgemm` and `vxcos` – where x denotes the respective label for real and complex cases). ScaLAPACK [15] routines are not involved.

### B. Standard diagonalization

The standard diagonalization scheme is entirely based on LAPACK [16] and ScaLAPACK routines for the serial and the parallel calculations, respectively. The scheme applied in WIEN2k (which will be called "full diagonalization" for the rest of the paper, since the entire matrix is handled and the eigenvalues are calculated in an exact way) is a 4-step procedure, which consists of the following routines – only the ScaLAPACK routines, both for real and complex matrices, are given:

- A Cholesky factorization of the overlap matrix $S$, using either `pdpotrf` or `pzpotrf`.
- The reduction of the general eigenvalue problem (see (8)) to standard form $H'C' = EC'$ with the routines `pdsygst` or `pzhegst`.
- The computation of the lowest eigenvalues (WIEN2k uses for routine calculations an energy cutoff as upper limit for routine calculations, and computes only about 10% of the $\epsilon_i$), using either `pdsyevr` or `pzheevx`.
- The back-transformation of the eigenvectors $C'$ to the original problem with the routines `pdtrsm` or `pztrsm`.

### C. Iterative diagonalization

In 2010 Blaha et al. [17] introduced an alternate eigensolver scheme that is much faster for bigger systems. The major simplification of this alternate scheme is based on the fact that the Kohn-Sham equations (4) have to be solved iteratively. Therefore, it is not necessary to solve them exactly for every step in the scf cycle. In addition the eigenvectors of the previous scf cycle provide a good starting point for the current solution because the change in the next iteration is expected to be small. An approximative solution is sufficient, as long as it converges at the same time as the scf cycle. The main steps of this procedure are the following (see Table I for example times):

- Calculation of estimated eigenvalues using the "old" eigenvectors from the last step, which is done by multiplying the eigenvectors with the Hamilton and overlap matrices $H$ and $S$ (`pdsymm` / `pzhemm`) and evaluating the resulting matrices (`pdsyr2k` / `pzher2k`) [17].
- Setup of a search space, which is done by performing a LU-factorization of $H$ (`pdgetrf` / `pzgetrf`) as the first step, then subsequently solving the resulting system of linear equations (`pdgetrs` / `pzgetrs`).
- Setup of a reduced eigenvalue problem with a dimension twice the number of eigenvalues, i.e. approximately 20% of the original matrix dimension (`pdgemm` / `pzgemm` and `pdsyr2k` / `pzher2k`).
- Solving of the reduced eigenvalue problem (as described in Section III-B).

Using this scheme one has to keep in mind that the time scales linearly with the number of calculated eigenvalues (see [17]), while for the full diagonalization the number of eigenvalues has only a small effect on the computational time.

## IV. Performance Evaluation

The performance of the described diagonalization schemes was evaluated on the third generation of the Vienna Scientific Cluster (VSC3), which provides the following test environment:

- CPU: Intel Xeon E5-2650v2, 2.60 GHz, 8 cores, 20 MB cache – two CPUs per node.
- Network: Infiniband (QDR-80) in a fat tree.
- Relevant software: Linux (Scientific Linux 6.6.), Intel MKL 11.3 (containing LAPACK 3.5.0 and ScaLAPACK 2.0.2), Intel Fortran Compiler 16.0.1, Intel MPI 5.1.

The input data used for the tests were derived from two different structures ($LiBH_4$ and $LiNH_2$) and represent realistic electronic structure problems which are routinely solved using WIEN2k. Different basis set sizes were used for testing purposes. Both real and complex eigenvalue problems (of different sizes) were solved, but here only one representative example of a relatively small real test-case (matrix dimension 24920 and 1620 computed eigenvalues) will be presented, which shows some interesting features of the current scaling behavior of the eigensolvers in WIEN2k. All tests have been repeated 4 times. The reported wall-clock times were usually reproducible within 3% and were taken from calls to a WIEN2k-specific timing-routine (based on the c-routine `walltime`), that is used to routinely instrument `lapw1` (see Table I). All speedups were related to the computation times on two cores – relating the times to serial results is not possible because of memory constraints (locally the upper limit for the matrix dimension is 20000) and the fact that different algorithms are used.

TABLE I

TIMES OF DIFFERENT STEPS OF THE WIEN2K SUBPROGRAM `lapw1` - DEALING WITH THE SETUP AND THE SOLUTION OF A GENERALIZED EIGENVALUE PROBLEM (SEE SECTION III FOR DETAILS). THE DIMENSION OF THE INVOLVED MATRICES IN THIS EXAMPLE WAS 24920X24920. 1620 EIGENVALUES WERE COMPUTED WITH THE ITERATIVE DIAGONALIZATION SCHEME. GIVEN ARE WALL-CLOCK TIMES IN SECONDS.

| | **Number of cores** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Task** | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 192 | 256 | 320 | 384 | 448 | 512 |
| | **Total time spent in program `lapw1`** | | | | | | | | | | | | |
| Full | 3581.6 | 2123.8 | 1172.0 | 728.1 | 409.3 | 254.7 | 253.4 | 201.2 | 88.9 | 92.6 | 87.6 | 82.2 | 73.6 |
| Iterative | 1486.9 | 821.0 | 426.0 | 245.8 | 123.8 | 76.4 | 43.1 | 34.3 | 29.9 | 27.6 | 26.7 | 24.7 | 25.3 |
| | **Setup times (same for both procedures)** | | | | | | | | | | | | |
| $H_{sp}$, $S_{sp}$ | 496.3 | 257.7 | 126.6 | 66.1 | 33.0 | 16.7 | 10.0 | 6.7 | 5.1 | 4.1 | 3.7 | 3.2 | 2.8 |
| $H_{nsp}$, $S_{nsp}$ | 350.5 | 194.7 | 103.2 | 71.9 | 30.9 | 17.1 | 7.8 | 5.7 | 4.6 | 3.8 | 3.2 | 2.8 | 2.5 |
| | **Sub-step times for full diagonalization** | | | | | | | | | | | | |
| Total | 2729.2 | 1667.0 | 938.4 | 586.5 | 342.0 | 217.6 | 232.3 | 185.5 | 75.9 | 81.4 | 77.3 | 72.9 | 64.9 |
| Cholesky | 154.8 | 85.4 | 44.1 | 25.8 | 17.4 | 11.9 | 5.5 | 4.9 | 3.9 | 3.4 | 2.9 | 2.6 | 2.0 |
| Reduction | 665.5 | 396.2 | 211.6 | 123.8 | 90.9 | 57.8 | 47.8 | 44.4 | 35.1 | 36.3 | 38.1 | 32.8 | 32.1 |
| Eigenvalues | 1899.1 | 1177.3 | 678.5 | 434.5 | 232.0 | 146.7 | 178.1 | 135.9 | 36.6 | 41.5 | 36.2 | 37.2 | 30.5 |
| | **Sub-step times for iterative diagonalization** | | | | | | | | | | | | |
| Total | 635.3 | 368.9 | 191.0 | 71.8 | 56.2 | 39.0 | 21.5 | 18.1 | 16.3 | 15.9 | 15.7 | 14.8 | 15.9 |
| Estimate for eigenvalues | 145.6 | 90.5 | 48.9 | 26.2 | 13.8 | 11.9 | 4.2 | 3.1 | 3.5 | 2.2 | 2.0 | 1.7 | 1.7 |
| LU-Factorization | 278.3 | 144.6 | 75.6 | 39.2 | 23.0 | 13.2 | 9.5 | 7.3 | 6.4 | 6.5 | 6.4 | 6.2 | 6.2 |
| Solution linear equations | 64.3 | 52.8 | 29.0 | 16.6 | 6.8 | 3.6 | 1.7 | 1.3 | 1.0 | 0.8 | 0.8 | 0.7 | 0.6 |
| Setup reduced problem | 87.5 | 53.3 | 28.6 | 15.7 | 8.1 | 6.7 | 2.5 | 1.8 | 2.0 | 1.3 | 1.1 | 1.0 | 1.0 |
| Solution reduced problem | 28.8 | 20.2 | 4.0 | 2.5 | 2.2 | 2.0 | 2.0 | 1.8 | 1.5 | 1.7 | 1.8 | 1.6 | 1.8 |

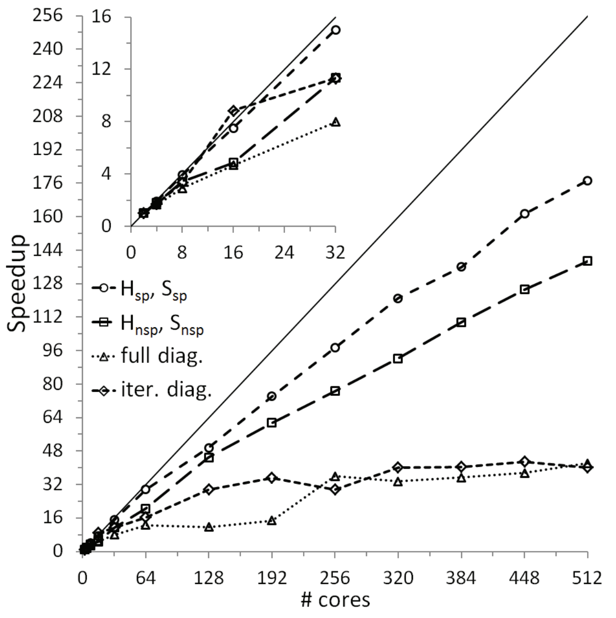

Fig. 1. Speedup for the main parts of `lapw1` (see Section III) computing a real matrix of dimension 24920. The inset shows a magnification for 0 to 32 cores.

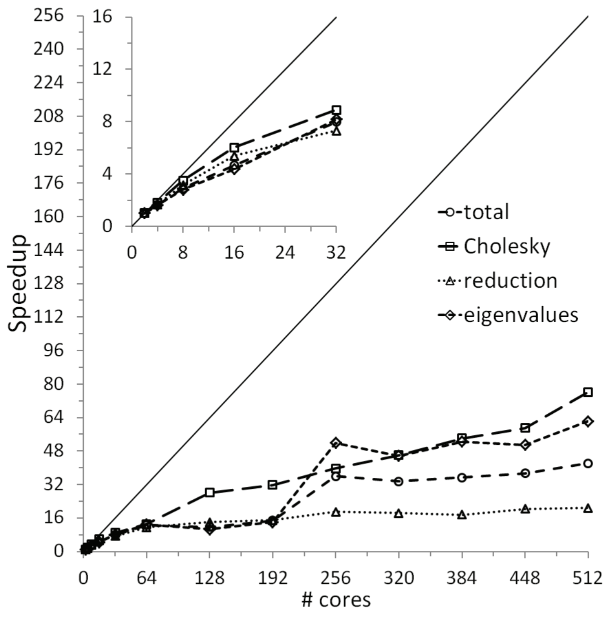

Fig. 2. Detailed speedup for the sub-steps of the full diagonalization of a matrix of dimension 24920. The back-transformation is omitted. A magnification for 0 to 32 cores is shown in the inset.

## V. DISCUSSION

As can be seen from the timings listed in Table I the setup time ($H_{sp}$,$S_{sp}$ and $H_{nsp}$,$S_{nsp}$) takes about 850 s on 2 cores, which is only about 30% of the time spent in full diagonalization (2730 s), but is even larger then the time spent in iterative diagonalization (635 s). If these ratios are compared to the timing with 512 cores (5, 65 and 16 s, respectively) it is obvious that the setup time scales much better with the number of cores then the diagonalization, changing it from an important (dominating for iterative diagonalization) to a minor part. The corresponding speedups are shown in Figure 1. The setup steps $H_{sp}$,$S_{sp}$ and $H_{nsp}$,$S_{nsp}$ show reasonable scaling – with the setup of the spherical matrix elements scaling almost linearly up to 96 cores – while the scaling of the total times of the eigensolver breaks down almost immediately (already at 64 cores only roughly 50% of the linear speedup is achieved).

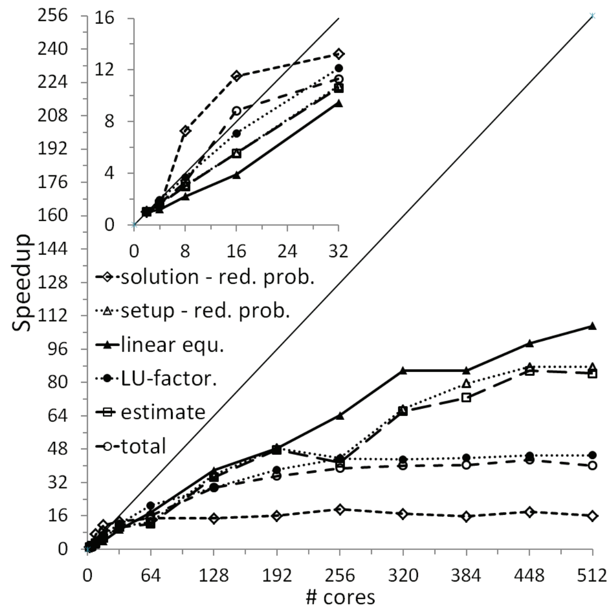

Fig. 3. Detailed speedup for the sub-steps of the iterative diagonalization scheme used to find the lowest 1620 eigenvalues of a matrix of dimension 24920. The inset shows a magnification for 0 to 32 cores.

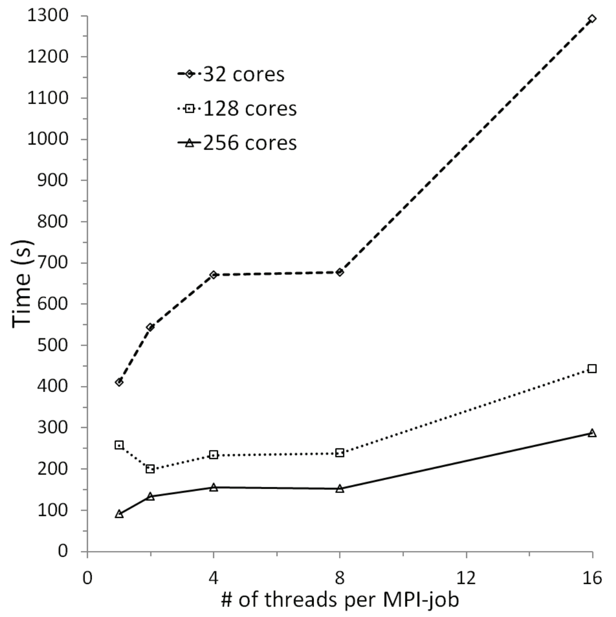

Fig. 4. Total wall-clock times (in seconds) for `lapw1` with the full diagonalization scheme depending on the number of openMP threads within a MPI-job.

This behavior also obviously determines the scaling of the total time in the case of the full diagonalization, while a better scaling for the iterative scheme can be seen, which derives from the larger contribution of the setup time. However, both schemes cannot be used in a meaningful way for matrices of the given size on more than 256 cores – usage of more than 256 cores did not yield a significant improvement in wall-clock time. Tests with bigger matrices (of dimensions 60000 and 73000 respectively) show that usage of 512 cores may be reasonable for those bigger cases.

The detailed speedups in Figures 2 and 3 show that the scaling of all sub-steps immediately breaks down on more than one node (32 cores and beyond) which is most likely caused by slower inter-node communication (via the network). Although some tasks (namely the Cholesky transformation in the full diagonalization scheme, and the eigenvalue estimation, the solution of the linear equation system, and the setup of the reduced eigenvalue problem) still show an upward trend on 512 cores, the speedup is low at below 50%. More importantly, the steps that dominate in terms of time (the eigenvalue calculation and the LU-factorization) do not yield significantly faster times beyond 128 cores.

The eigenvalue computation (`pdsyevr`) in the full diagonalization shows a distinct performance irregularity between 128 and 192 cores (the speedup even drops for 128 cores), which is reflected in the total time (see Table I and Figure 1), because this is the dominant sub-step of this scheme for smaller number of cores. For 256 cores the performance of this algorithm improves drastically and shows a considerably better scaling than the reduction step (`pdsygst`), however, there is no significant improvement beyond 256 cores. This is most likely due to an unfavorable rectangular matrix decomposition at 128 cores (16x8). A test with additional openMP-parallelism showed that this problem can be (slightly) reduced if one uses 2 threads within each MPI-job. However, this approach cannot be used in general, because a time reduction was found only for 128 cores (as depicted in Figure 4), which is most likely due to the fact that 2 threads in this case lead to a possible square decomposition of the matrix (8x8). The tests for 32 and 256 cores show a substantial slowdown for larger numbers of threads.

The strange behavior for the solution of the reduced eigenvalue problem during the iterative diagonalization (see Figure 3 inset) occurs due to the smallness of the involved matrices – if enough cores are used the whole problem can be kept in the cache and, therefore, be solved without any memory access due to cache misses. This means that the time of this step will remain more or less constant upon increasing the number of cores. Furthermore the performance drops significantly both for the eigenvalue estimations and the setup of the reduced eigenvalue problem (both `pdsyr2k`) on 256 cores (see Figure 3). This may be connected to the square decomposition of the matrix at this number of cores (16x16), since a similar, but less pronounced performance drop occurs on 64 cores for both steps.

## VI. Conclusion

The setup of the Hamilton matrix $H$ and the overlap matrix $S$ scales reasonably well up to 1024 cores for the considered problem size (matrix dimensions of 20000-70000). The trend suggests a drop in speedup beyond that number of cores. However, this is currently not a problem, since the usage of more than 1024 cores is not sensible anyway and the contribution of the setup to the total time is only minor,

The time-critical step of the full diagonalization scheme is clearly the determination of the eigenvalues. Moreover, a significant performance irregularity was observed around 128 cores (that is less pronounced for complex matrices), which is most likely connected to unfavorable matrix decomposition. This could be slightly improved using openMP/MPI-parallelism for 128 cores. Using openMP-parallelism in general is not useful, however, since depending on the core number the inverse effect may occur. The iterative diagonalization scales considerably better than the full diagonalization between 32 and 256 cores (albeit still within only 25 to 50% of the ideal speedup), in addition to the faster absolute times. Beyond 256 cores both schemes stagnate at a comparable (low) level for small cases (doubling the used cores yielded no improvement in time). For larger systems 512 cores can still be used reasonably well, however, a number beyond that does not improve the time significantly. It is thus important to adjust the maximum number of cores to the specific problem size.

In order to using more cores in a meaningful way this diagonalization bottleneck in the eigensolver step needs to be improved upon. As a first approach to do this, ELPA [18] will be tested as possible substitute for the ScaLAPACK routines.

## Acknowledgment

This publication is based upon work supported by the VSC Research Center funded by the Austrian Federal Ministry of Science, Research, and Economy (bmwfw).
The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

## References

[1] K. Schwarz, P. Blaha, and S. B. Trickey, “Electronic structure of solids with WIEN2k,” *Mol. Phys.*, vol. 108, pp. 3147–3166, 2010.

[2] P. Koehl and M. Levitt, “A brighter future for protein structure prediction,” *Natures Struct. Biol.*, vol. 6, pp. 108–111, 1999.

[3] A. Warshel, S. P. K., M. Kato, and P. W. W., “Modeling electrostatic effects in proteins,” *Biochim. Biophys. Acta*, vol. 1764, pp. 1647–1676, 2006.

[4] J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*. London: Methuen, 1975, ISBN: 0-416-52340-4.

[5] K. Binder, *The Monte Carlo Method in Condensed Matter Physics*. New York: Springer, 1995, ISBN: 0-387-54369-4.

[6] P. Hohenberg and W. Kohn, “Inhomogeneous electron gas,” *Phys. Rev.*, vol. 136, B864–B871, 1964.

[7] W. Kohn and L. J. Sham, “Self-consistent equations including exchange and correlation effects,” *Phys. Rev.*, vol. 140, A1133–A1138, 1965.

[8] P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka, and J. Luitz, *WIEN2k: An Augmented Plane Wave plus Local Orbitals Program for Calculating Crystal Properties*. Austria: Vienna University of Technology, 2001.

[9] R. Laskowski and P. Blaha, “*Ab initio* study of h-BN nanomeshes on Ru(001), Rh(111) and Pt(111),” *Phys. Rev. B.*, vol. 81, p. 075 418, 2010.

[10] R. Bliem, E. McDermott, P. Ferstel, M. Setvin, O. Gamba, J. Pavelec, M. A. Schneider, M. Schmid, U. Diebold, P. Blaha, L. Hammer, and G. S. Parkinson, “Subsurface cation vacancy stabilization of the magnetite (001) surface,” *Science*, vol. 346, pp. 1215–1218, 2014.

[11] R. Laskowski and P. Blaha, “Understanding of $^{33}$S NMR shielding in inorganic sulfides and sulfates,” *J. Phys. Chem. C*, vol. 119, pp. 19 390–19 396, 2015.

[12] M. Born and R. Oppenheimer, “Zur quantentheorie der molekeln,” *Ann. Phys.*, vol. 84, pp. 457–484, 1927.

[13] A. J. Cohen, P. Mori-Sánchez, and W. Yang, “Challenges for density functional theory,” *Chem. Rev.*, vol. 112, pp. 289–320, 2012.

[14] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumdsdaine, A. Petitet, R. Pozo, K. Remnington, and R. C. Whaley, “An updated set of basic linear algebra subprograms (BLAS),” *ACM Trans. Math. Soft.*, vol. 28-2, pp. 135–151, 2002.

[15] L. S. Blackford, J. Choi, A. Cleary, E. D’Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users’ Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997, ISBN: 0-89871-397-8 (paperback).

[16] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users’ Guide*, Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999, ISBN: 0-89871-447-8 (paperback).

[17] P. Blaha, H. Hofstätter, O. Koch, R. Laskowski, and K. Schwarz, “Iterative diagonalization in augmented plane wave based methods in electronic structure calculations,” *J. Comput. Phys.*, vol. 229, pp. 453–460, 2010.

[18] T. Auckenthaler, V. Blum, H.-J. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, and P. R. Willems, “Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations,” *Parallel Comput.*, vol. 37, pp. 783–794, 2011.