



ELSEVIER

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations [☆]

T. Auckenthaler^a, V. Blum^b, H.-J. Bungartz^a, T. Huckle^a, R. Johanni^c, L. Krämer^d, B. Lang^{d,*},
H. Lederer^c, P.R. Willems^d

^aFakultät für Informatik, Technische Universität München, D-85748 Garching, Germany

^bFritz-Haber-Institut der Max-Planck-Gesellschaft, D-14195 Berlin, Germany

^cRechenzentrum Garching der Max-Planck-Gesellschaft am Max-Planck-Institut für Plasmaphysik, D-85748 Garching, Germany

^dFachbereich C, Bergische Universität Wuppertal, D-42097 Wuppertal, Germany

ARTICLE INFO

Article history:

Available online xxx

Keywords:

Electronic structure calculations
Eigenvalue and eigenvector computation
Blocked Householder transformations
Divide-and-conquer tridiagonal eigensolver
Parallelization

ABSTRACT

The computation of selected eigenvalues and eigenvectors of a symmetric (Hermitian) matrix is an important subtask in many contexts, for example in electronic structure calculations. If a significant portion of the eigensystem is required then typically direct eigensolvers are used. The central three steps are: reduce the matrix to tridiagonal form, compute the eigenpairs of the tridiagonal matrix, and transform the eigenvectors back. To better utilize memory hierarchies, the reduction may be effected in two stages: full to banded, and banded to tridiagonal. Then the back transformation of the eigenvectors also involves two stages. For large problems, the eigensystem calculations can be the computational bottleneck, in particular with large numbers of processors. In this paper we discuss variants of the tridiagonal-to-banded back transformation, improving the parallel efficiency for large numbers of processors as well as the per-processor utilization. We also modify the divide-and-conquer algorithm for symmetric tridiagonal matrices such that it can compute a subset of the eigenpairs at reduced cost. The effectiveness of our modifications is demonstrated with numerical experiments.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Finding the eigenvalues and eigenvectors of symmetric (Hermitian) matrices is a classic problem in linear algebra, and computationally efficient solutions are enormously important. Since the solution effort for the full spectrum scales as $O(n^3)$ with matrix size n , solving an eigenproblem as part of a larger computational task can easily dominate the entire workload, or can even render the solution impossible even on the most advanced computational hardware available today. Here we discuss two particular improvements to the efficiency of general-purpose eigensolvers, regarding the overall workload and the parallel performance on platforms with thousands of CPUs. The examples in the paper arise in large-scale, all-electron electronic structure theory, i.e., the prediction of materials properties from the atomic scale on upwards, based only on the “first principles” of quantum mechanics. However, all algorithms described in this paper are general, and applicable in any field where the solution of large (almost) dense eigenproblems is needed.

[☆] This work was supported by the Bundesministerium für Bildung und Forschung within the project “ELPA—Hochskalierbare Eigenwert-Löser für Petaflop-Großanwendungen”, Förderkennzeichen 01IH08007.

* Corresponding author.

E-mail addresses: auckenth@in.tum.de (T. Auckenthaler), blum@fhi-berlin.mpg.de (V. Blum), bungartz@in.tum.de (H.-J. Bungartz), huckle@in.tum.de (T. Huckle), rrj@rzg.mpg.de (R. Johanni), lkraemer@math.uni-wuppertal.de (L. Krämer), lang@math.uni-wuppertal.de (B. Lang), lederer@rzg.mpg.de (H. Lederer), willems@math.uni-wuppertal.de (P.R. Willems).

A central task of electronic structure theory is the solution of Schrödinger-like eigenproblems $\hat{H}\Psi_m = E_m\Psi_m$, with \hat{H} a Hamilton operator, Ψ_m a wave function and E_m the corresponding eigenenergy. Most frequently the ground-state total energy of a many-electron system in a given external potential (usually due to classical atomic nuclei at fixed geometry) is sought. The most widely used approach to this problem is Kohn–Sham (KS) density functional theory (DFT) [1], for which a set of effective single-particle eigenvalue/eigenfunction pairs for an eigenproblem $\hat{H}_{\text{KS}}\psi_i = \epsilon_i\psi_i$ must be found. Similar eigenproblems arise in many other contexts in electronic structure theory, for example Hartree–Fock (HF) theory [2], or in the description of optical excitations through the Bethe–Salpeter [3] or Casida [4] equations. In KS-DFT, $\psi_i(\mathbf{r})$ are continuous functions in three-dimensional space. In practice, they are usually discretized by inserting a set of basis functions $\{\phi_i(\mathbf{r}), i = 1, \dots, n\}$ such that $\psi_i(\mathbf{r}) = \sum_j c_{ij}\phi_j(\mathbf{r})$. The result is a *generalized* matrix eigenproblem:

$$\mathbf{H}_{\text{KS}}\mathbf{c}_i = \epsilon_i\mathbf{S}\mathbf{c}_i, \quad (1)$$

with Hamilton matrix \mathbf{H}_{KS} and overlap matrix \mathbf{S} , where $\mathbf{S} = \mathbf{I}$ for some but not all basis types. In KS or HF theory, only the lowest k out of n possible eigenvalue/-vector pairs are needed. They correspond to those electronic “orbitals” which are “occupied,” and which thus contribute to the ground-state electron density $n(\mathbf{r}) = \sum_{i=1}^k |\psi_i(\mathbf{r})|^2$. Eq. (1) is not a linear eigenvalue problem as the Hamiltonian $\mathbf{H}_{\text{KS}} \equiv \mathbf{H}_{\text{KS}}[n(\mathbf{r})]$ depends on its own eigenvectors \mathbf{c}_i through $n(\mathbf{r})$. One thus seeks a particular Hamiltonian that yields as solutions the same eigenvectors \mathbf{c}_i used to construct it in the first place—a “self-consistent” solution of (1). To find such solutions \mathbf{H}_{KS} and \mathbf{c}_i iteratively, (1) must be solved successively for ten(s) of times even in simple cases such as self-consistency for a single, fixed nuclear geometry. Hundreds of thousands (or more) iterations may be desirable for other questions, such as long *ab initio* molecular dynamics simulations, where the nuclear geometry changes with every step.

The key point is that the solution effort to find \mathbf{c}_i scales as $O(N^3)$. Here, N loosely refers to overall *system* size, i.e., the number and type of atoms in a molecule or solid, not necessarily just the matrix size n . This scaling arises either through the matrix problem (1) itself, or at least through the practical need to orthonormalize \mathbf{c}_i . By contrast, all other steps needed to set up and solve \mathbf{H}_{KS} can be implemented to scale essentially as $O(N)$. Even if these steps can dominate for many cases and require sophisticated approaches of their own (e.g., parallel FFTs), the solution of (1) will eventually come to dominate as the system size increases. In practice, the crossover often occurs for molecule or unit cell sizes of ≈ 100 – 1000 s of atoms [5–10] even for favorable basis sets, well within the range of interest for many applications (biomolecules, nano-electronics, defects in solids, etc.). In the limit, $O(N^3)$ scaling means that any *larger* calculations are rendered infeasible even on the most advanced supercomputers, unless the prefactor of the time needed for the respective operations can be reduced. Adding to the problem is the fact that solving (1) directly as a linear problem in parallel requires quite fine-grained, communication-intensive operations. Especially on massively parallel computer systems with thousands of processor cores and using standard implementations, this places an additional practical limitation on the calculations that can be done.

Solving or even circumventing the solution of (1) is thus an active research field, with many new and original contributions even in the most recent literature [6,7,9,11–19]. Among the strategies pursued in electronic structure theory, one finds:

- (i) For specific problem classes—eigenspectra with a gap between occupied and unoccupied electronic states—the solution of (1) can be avoided altogether; see, e.g., [5,20–23]. Where this works, $O(N)$ scaling for systems up to even millions of atoms has been demonstrated [24]. However, $O(N)$ strategies are not generally applicable. For example, the requirement for a spectral gap excludes all metals. Likewise, any calculation where the effective single-particle band structure ϵ_i itself is of interest of course requires a solution of (1).
- (ii) Instead of iterating exact solutions of the linear problem (1) for successive intermediate versions of \mathbf{H}_{KS} , one can view the full Kohn–Sham problem directly as a non-linear minimization problem with parameters \mathbf{c}_i . Refined minimization strategies (often, conjugate gradients) can be used to find \mathbf{c}_i . (Ref. [16] gives a good overview over the literature).
- (iii) Another active research area are *iterative* solution strategies for the linear problem (Lanczos, Davidson [25]-based, locally optimal preconditioned gradient [26,27], many others [28,29]), for cases where only a small fraction k/n of all possible eigenpairs are needed (e.g., large, systematically convergeable basis types such as plane waves or real-space grid functions). For example, a well preconditioned blocked Davidson scheme is superior even for augmented plane wave based all-electron DFT, with $k/n \approx 3$ – 10% [9]. To our knowledge, iterative strategies such as (ii) and (iii) usually still require at least one $O(N^3)$ operation, at least in practice [14,6,27,29,30,9] (subspace diagonalization, re-orthonormalization, matrix inversion at the outset, etc.).
- (iv) A straightforward path to push the crossover point between (1) and all other, $O(N)$ type operations out is to reduce n , the basis size itself. Obviously, in a system with k occupied states to be treated, $n \geq k$ sets a lower bound. Nonetheless, k/n is often already in the range 10%–40% for localized Gaussian-type basis sets of quantum chemistry [2,6]. Other choices, e.g., Slater-type or numerically tabulated atom-centered orbitals (NAOs) (see [8] for references and discussion), can yield essentially converged accuracy with even more compact basis sets that still remain generically transferable. Finally, a recent localization-based filtering approach [7,19] contracts a large, generic basis set into a system-dependent *minimal* basis ($n = k$), prior to solving (1). The $O(N^3)$ bottleneck is then reduced to the minimum cost $O(k^3)$, at negligible accuracy loss.

In practice, (iv) is attractive because, for many relevant system sizes, (1) then does not yet dominate the overall computational effort. Therefore, robust, conventional eigensolver strategies such as implemented in LAPACK [31] can be used,

avoiding system-specific complications such as the exact form of the eigenspectrum, or the choice of an optimal preconditioning strategy [11,9]. Even for (i)–(iii), though, a conventional diagonalization of some kind may still be required or is a necessary fallback.

In general, the solution of (1) proceeds in five steps: (A) Transformation to a dense standard eigenproblem (e.g., by Cholesky decomposition of \mathbf{S}), $\mathbf{H}_{\text{KS}}\mathbf{c}_l = \epsilon_l\mathbf{S}\mathbf{c}_l \rightsquigarrow \mathbf{A}\mathbf{q}_A = \lambda\mathbf{q}_A$, $\lambda \equiv \epsilon_l$; (B) Reduction to tridiagonal form, $\mathbf{A} \rightsquigarrow \mathbf{T}$; (C) Solution of the tridiagonal problem for k eigenvalues and vectors, $\mathbf{T}\mathbf{q}_T = \lambda\mathbf{q}_T$; (D) Back transformation of k eigenvectors to dense orthonormal form, $\mathbf{q}_T \rightsquigarrow \mathbf{q}_A$; (E) Back transformation to the original, non-orthonormal basis, $\mathbf{q}_A \rightsquigarrow \mathbf{c}_l$. Fig. 1 shows the overall timings of these operations on a massively parallel IBM BlueGene/P system, for one specific example: the electronic structure of a 1003-atom polyaniline peptide (small protein) conformation in an artificially chosen, fixed α -helical geometry. The example is set up using the “Fritz Haber Institute *ab initio* molecular simulations” (FHI-aims) all-electron electronic structure package [8,32], at essentially converged basis set accuracy for DFT (*tier 2* [8]). For (1), this means $n = 27,069$. The number of calculated eigenpairs is $k = 3410$, somewhat more than the theoretical minimum $k_{\text{min}} = 1905$, one state per two electrons. Steps (A)–(E) were performed using only subroutine calls as in the ScaLAPACK [33] library where available, as implemented in IBM’s system-specific ESSL library, combined as described briefly in [8, Section 4.2]. The reason is that ScaLAPACK or its interfaces are widely used for (massively) parallel linear algebra and readily available; no claim as to whether our use is the best or only possible alternative is implied. ScaLAPACK provides the driver routine `pdsyevd`, which calls `pdsytrd`, `pdstede`, and `pdormtr` for tridiagonalization, solution of the tridiagonal eigenproblem and back transformation respectively. `pdstede` is based on the divide-and-conquer (D&C) algorithm, tridiagonalization and back transformation are done using Householder transformations and blocked versions thereof [34,35]. The back transformation was done only for the needed eigenvectors.

Our point here are some key conclusions, in agreement with reports in the wider literature [12,6,36]. What is most apparent from Fig. 1 is that even for this large electronic structure problem, the calculation does not scale beyond 1024 cores, thus limiting the performance of any full electronic structure calculation with more processors. By timing steps (A)–(E) individually, it is obvious that (B) the reduction to tridiagonal form, and then (C) the solution of the tridiagonal problem using the D&C approach dominate the calculation, and prevent further scaling. For (B), the main reason is that the underlying Householder transformations involve matrix–vector operations (use of BLAS-2 subroutines and unfavorable communication pattern); the magnitude of (C) is more surprising (see below). By contrast, the matrix multiplication-based transformations (A), (D), and (E) either still scale or take only a small fraction of the overall time.

In the present paper, we assume that step (A) already has been completed, and step (E) will not be considered, either. We present a new parallel implementation based on the two-step band reduction of Bischof et al. [37] concerning step (B), tridiagonalization; Section 2.1, with improvements mainly for step (D), back transformation; Section 2.2. We also extend the D&C algorithm, thus speeding up step (C); Section 3. Some additional optimization steps in the algorithmic parts not specifically discussed here (reduction to banded form, optimized one-step reduction to tridiagonal form, and corresponding back transformations) will be published as part of an overall implementation in [38]. These routines are also included in recent production versions of FHI-aims. For simplicity we will present only the real symmetric case; the complex Hermitian case is similar.

In addition to synthetic testcases, we show benchmarks for two large, real-world problems from all-electron electronic structure theory: first, the $n = 27,069$, $k = 3410$ polyaniline case of Fig. 1, which will be referred to as Poly27069 problem in the following, and second, an $n = 67,990$ generalized eigenproblem arising from a periodic Pt (100)-“(5 × 40)”, large-scale reconstructed surface calculation with 1046 heavy-element atoms, as needed in [39]. In the latter calculation, the large fraction of core electrons for Pt (atomic number $Z = 78$) makes for a much higher ratio of needed eigenstates to overall basis size, $k = 43,409 \approx 64\%$, than in the polyaniline case, even though the basis set used is similarly well converged. This problem will be referred to as Pt67990. Benchmarks are performed on two distinct computer systems: The IBM BlueGene/P machine “genius” used in Fig. 1, and a Sun Microsystems-built, Infiniband-connected Intel Xeon (Nehalem) cluster with individual eight-core nodes. We note that for all standard ScaLAPACK or PBLAS calls, i.e., those parts not implemented by ourselves, the optimized ScaLAPACK-like implementations by IBM (ESSL) or Intel (MKL) were employed.

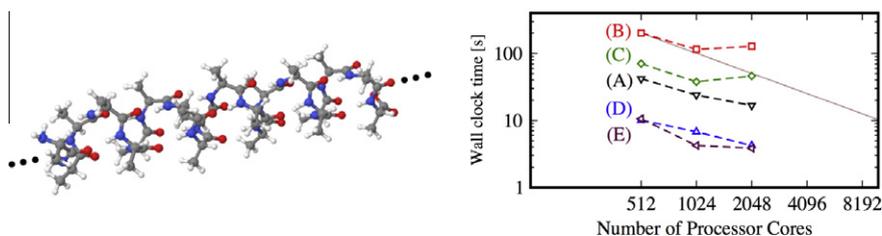


Fig. 1. Left: Segment of the α -helical polyaniline molecule Ala₁₀₀ as described in the text. Right: Timings for the five steps (A): reduction to standard eigenproblem, (B): tridiagonalization, (C): solution of the tridiagonal problem, and back transformation of eigenvectors to the full standard problem (D) and the generalized problem (E), of a complete eigenvalue/-vector solution for this molecule, $n = 27,069$, $k = 3410$, as a function of the number of processor cores. The calculation was performed on an IBM BlueGene/P system, using a completely ScaLAPACK-based implementation. Step (C) was performed using the divide-and-conquer method.

2. Efficient tridiagonalization and back transformation

The two-step band reduction of Bischof et al. [37] is a promising approach for an efficient parallel tridiagonalization of dense symmetric matrices. In this approach the full matrix \mathbf{A} of size n is first reduced to banded symmetric form with (semi) bandwidth b . In a second step the banded matrix \mathbf{B} is brought to tridiagonal form. This procedure has the advantage that the vast majority of the operations can be done with highly efficient BLAS-3 routines instead of memory bandwidth limited BLAS-2 routines. The drawback of the two-step approach is the additional computational effort for the second reduction step and its back transformation respectively. While the reduction from banded to tridiagonal form is comparatively cheap ($6bn^2$ flops), the additional costs for the back transformation are significant ($2kn^2$ flops). This holds especially true for a high number k of desired eigenvectors.

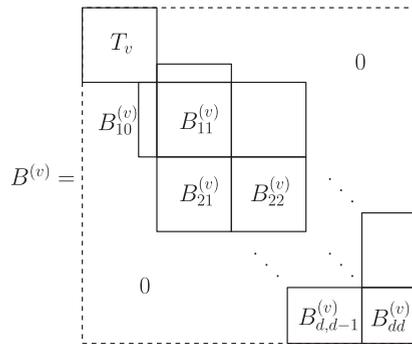
For the parallel reduction to banded form and the corresponding back transformation of eigenvectors we used well established algorithms based on Householder transformations. We refer to [37] and [40] for further information.

In Section 2.1 we will briefly revisit the bulge-chasing algorithm for reducing banded symmetric matrices to tridiagonal form [41]. In Sections 2.2 and 2.3 we will present new parallelization strategies for the tridiagonal-to-banded back transformation of eigenvectors.

2.1. Reduction from banded to tridiagonal form

For the reduction of banded matrices to tridiagonal form there exist specialized algorithms which exploit the banded structure of the matrix. In [42], Schwarz introduced an algorithm which successively decreases the bandwidth of the matrix by one, using Givens rotations. In [41] a parallel algorithm based on Householder transformations was presented, where $b - 1$ subdiagonal elements are removed at once. This concept was generalized to a successive band reduction [43,44]. All algorithms necessarily generate intermediate fill-in. To preserve the banded structure, all or some of the fill-in is removed with bulge-chasing techniques.

The bulge-chasing algorithm in [41] takes $n - 2$ stages to tridiagonalize a symmetric matrix $\mathbf{B} = (b_{ij}) =: \mathbf{B}^{(1)} \in \mathbb{R}^{n \times n}$ having (semi) bandwidth b . In the v th stage, column v of the band is brought to tridiagonal form. To this end, the remaining band is treated as a block tridiagonal matrix $\mathbf{B}^{(v)}$ with $\mathbf{B}_{00}^{(v)} = b_{vv} \in \mathbb{R}^{1 \times 1}$ and $\mathbf{B}_{10}^{(v)} = (b_{v+1,v}, \dots, b_{v+b,v})^T \in \mathbb{R}^{b \times 1}$ containing just the first remaining column, and diagonal blocks $\mathbf{B}_{\beta\beta}^{(v)}$ and subdiagonal blocks $\mathbf{B}_{\beta+1,\beta}^{(v)}$ of size $b \times b$ for $\beta \geq 1$ (except for smaller blocks at the end of the band):



Note that because of symmetry only the lower triangle of $\mathbf{B}^{(v)}$ needs consideration.

Stage v is initiated by a length- b Householder transformation, which reduces the first remaining column to tridiagonal form: $\mathbf{U}_1^{(v)T} \mathbf{B}_{10}^{(v)} = (*, 0, \dots, 0)^T$. This transformation must be applied to the whole first block row and block column, $\bar{\mathbf{B}}_{11} = \mathbf{U}_1^{(v)T} \mathbf{B}_{11}^{(v)} \mathbf{U}_1^{(v)}$, $\bar{\mathbf{B}}_{21} = \mathbf{B}_{21}^{(v)} \mathbf{U}_1^{(v)}$, thereby filling the subdiagonal block completely. To preserve the banded structure for future stages, at least the zeros in the first column of \mathbf{B}_{21} must be recovered with a second Householder transformation, and so on. More precisely, in the β th step of stage v , the first column of the subdiagonal block $\mathbf{B}_{\beta+1,\beta}^{(v)}$ has been filled by the preceding transformation, $\bar{\mathbf{B}}_{\beta,\beta-1} = \mathbf{B}_{\beta,\beta-1}^{(v)} \mathbf{U}_{\beta-1}^{(v)}$. The next length- b transformation $\mathbf{U}_{\beta}^{(v)}$ is chosen to re-eliminate this column, $\mathbf{U}_{\beta}^{(v)T} \bar{\mathbf{B}}_{\beta,\beta-1}(:, 1) = (*, 0, \dots, 0)^T$. This transformation must be applied to the whole β th block row and block column, $\bar{\mathbf{B}}_{\beta\beta} = \mathbf{U}_{\beta}^{(v)T} \mathbf{B}_{\beta\beta}^{(v)} \mathbf{U}_{\beta}^{(v)}$, $\bar{\mathbf{B}}_{\beta+1,\beta} = \mathbf{B}_{\beta+1,\beta}^{(v)} \mathbf{U}_{\beta}^{(v)}$, which in turn leads to the transformation $\mathbf{U}_{\beta+1}^{(v)}$. The stage is complete when the end of the band is reached, and then the block decomposition is shifted by one row/column for the next stage. (Thus the rows/columns affected by $\mathbf{U}_{\beta}^{(v+1)}$ are shifted by one position, as compared to those affected by $\mathbf{U}_{\beta}^{(v)}$.)

The transformations from the whole reduction and their data dependency are sketched in Fig. 2. One column (v) of the figure contains the Householder vectors (their nonzero entries) from one stage v of the described algorithm. The first sweep of transformations (vectors painted white) eliminates elements of the original banded matrix \mathbf{B} . All other sweeps (painted in gray and black) remove intermediate fill-in.

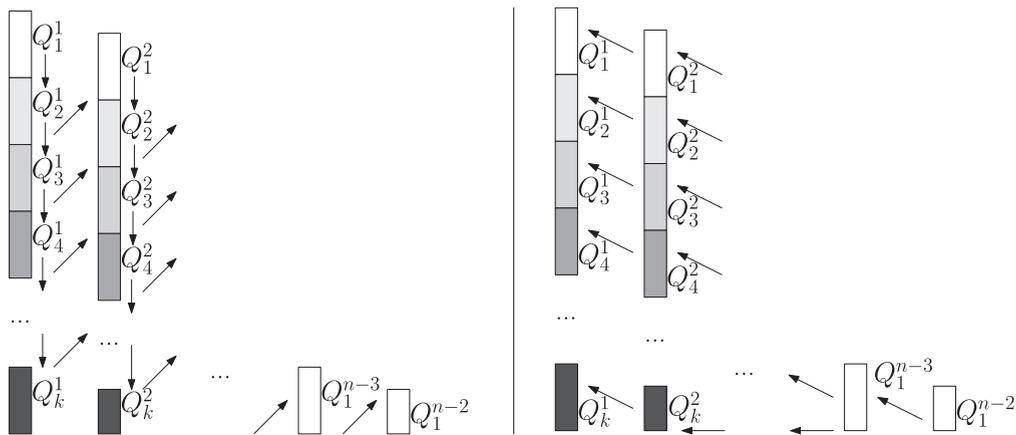


Fig. 2. Householder vectors from the bulge-chasing algorithm for the reduction from banded to tridiagonal form [41]. The arrows indicate the order of execution during reduction (left) and back transformation of eigenvectors (right).

2.2. Tridiagonal-to-banded back transformation of eigenvectors

In the tridiagonal-to-banded back transformation all the transformations from the banded-to-tridiagonal reduction have to be applied from the left side to the desired eigenvectors \mathbf{q}_i of the tridiagonal matrix,

$$\mathbf{q}_{Bi} = \left(\prod_{v,\beta} \mathbf{U}_\beta^{(v)} \right) \mathbf{q}_{Ti}, \quad i = 1, \dots, k. \quad (2)$$

For this task two different approaches have been implemented, which rely on two different parallel data layouts. Of course, the Householder transformations are not given as matrices, but as a set of Householder vectors \mathbf{v} , where each vector has at most b nonzero elements: $\mathbf{U}_\beta^{(v)} = \mathbf{I} - \tau \mathbf{v} \mathbf{v}^T$.

The work from (2) can run perfectly in parallel, because each eigenvector can be transformed independently. This fact has been used for the first parallelization approach, where the k eigenvectors are distributed uniformly across the p processes. For the second approach [45] the k eigenvectors are seen as a matrix of size $n \times k$, which is distributed in a 2D blocked manner across a 2D processor grid with p_r rows and p_c columns. Each process transforms its local part of the matrix. In the following, the costs of the two approaches will be analyzed and compared to each other. According to the data distribution, the approaches will be called 1D- and 2D-parallelization.

For the 1D-approach each process transforms k/p eigenvectors. Thus the computational costs are of order $2kn^2/p$. Additionally every process needs all the Householder vectors from the reduction step, leading to a communication volume of $n^2/2$ words per process. Beside the distribution of Householder vectors no further synchronization is necessary.

For the 2D-parallelization a second level of parallelism is used. A Householder transformation alters only those rows of the eigenvector matrix where the corresponding Householder vector has nonzero entries. This fact allows us to transform a single eigenvector in parallel. The 2D-approach is organized as follows: The k eigenvectors are distributed uniformly across p_c process columns, similarly to the 1D-approach. The individual eigenvectors in turn are distributed in a blocked manner across the p_r processes of a process column. For the parallelization within one process column the dependencies in Fig. 2 (right) have to be preserved. This leads to a pipelining algorithm. The process at the bottom of a process column starts the pipeline and applies the Householder transformations from the first sweep to its local part of the eigenvectors. In the next step the upper neighboring process can apply transformations from sweep one to its local part of the eigenvector

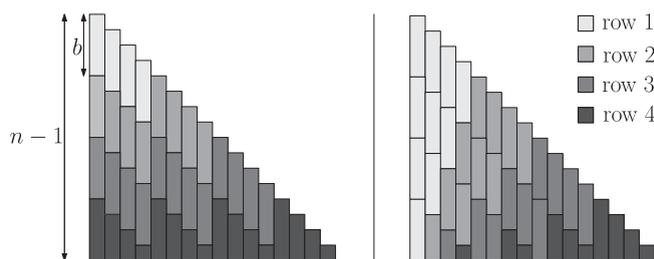


Fig. 3. Example of the distribution of Householder vectors with static (left) and dynamic (right) data distribution ($n = 17$, $b = 4$). The Householder vectors with the same shading are distributed across one row of the 4×4 process grid.

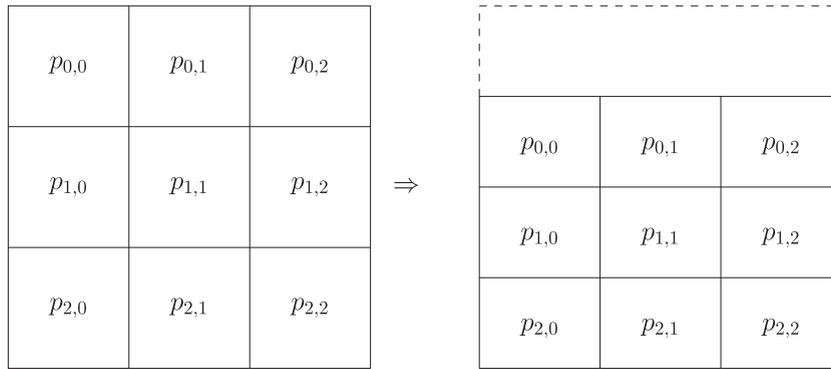


Fig. 4. Dynamic adaptation of the 2D data layout.

Table 1

Comparison of different parallelization approaches for the back transformation of eigenvectors (costs per process).

	Computation [flops]	Communication [#words]
1D	$2kn^2/p$	$n^2/2$
2D without load balancing	$4kn^2/p$	$n^2/p_r + 2kn/p_c$
2D	$2kn^2/p$	$n^2/2p_r + 3kn/p_c$

matrix, while the process on the bottom can start with the transformations from sweep two, and so on. After each of the n/b steps, vertically neighboring processes have to exchange blocks of size $b \times (k/p_c)$ due to data dependencies. But in contrast to the 1D-approach, the communication scales with the number of processes (1D: $n^2/2$, 2D: n^2/p_r words).

The distribution of Householder vectors is adapted to the distribution of work. Each Householder vector resides in the process row where it will be used during the algorithm. An example of the Householder vector distribution is depicted in Fig. 3. The picture also reveals the poor load balancing of the 2D-parallelization. The lower blocks of the eigenvector matrix are affected by much more Householder transformations than the upper blocks. This leads to load imbalances by a factor of 2 and a computation amount of $4kn^2/p$ for the lowermost processes.

The load imbalances have been resolved with a dynamic adaptation of the parallel data layout (Fig. 4). After each sweep, the b uppermost rows are removed from the matrix and the data distribution is adapted accordingly. To avoid memory imbalances, the removed rows have to be distributed uniformly to the whole processor grid, which results in additional communication costs of size kn/p_c for the topmost processes. On the other hand the dynamic data layout not only balances the computational load, but also the distribution of Householder vectors (see Fig. 3). Table 1 summarizes the costs for the different parallelization strategies.

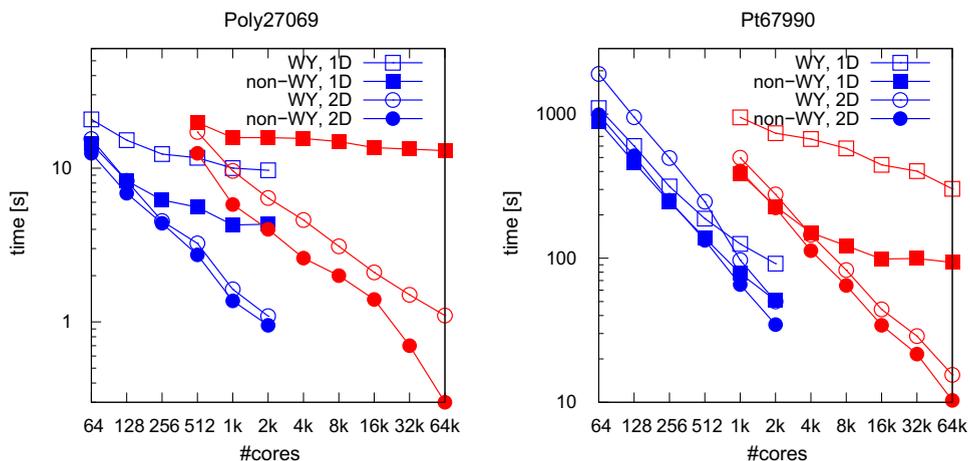


Fig. 5. Strong scalability of the tridiagonal-to-banded back transformation of eigenvectors for Poly27069 and Pt67990 (bandwidth $b = 64$). Lines starting at 64 cores: Intel cluster, lines starting at 512 or 1024 cores: BlueGene/P. The WY, 1D times for Poly27069 on BlueGene/P are above the plotting area.

Fig. 5 shows the strong scalability of the tridiagonal-to-banded back transformation for the Poly27069 and Pt67990 problems on the BlueGene/P and the Intel Nehalem cluster. For the Pt67990 problem, both parallelization strategies perform well, because most time is spent in computations. For the Poly27069 problem, however, the 1D parallelization does not scale beyond 1024 cores on the BlueGene/P and much less than ideal beyond 256 cores on the Intel Nehalem cluster. The different variants of the algorithm (WY, non-WY) and their behavior will be described in the next subsection.

2.3. Non-WY approaches for blocking Householder transformations

The back transformation of eigenvectors can be performed using WY transformations or variants thereof; cf. e.g. [34,35]. Here n_b Householder transformations are combined to a blocked Householder transformation, which is represented by matrices W and Y (Fig. 6). While the application of blocked Householder transformations is very efficient due to the use of BLAS-3 routines, it also creates a lot of computational overhead compared to the non-blocked application of the transformations. Each Householder vector has b nonzero entries. Thus, the non-blocked application of n_b such Householder transformations to k eigenvectors requires $4kbn_b$ flops. Due to the fact that the nonzero entries are shifted in each Householder vector, the matrices W and Y are not of size $b \times n_b$ but $(b + n_b - 1) \times n_b$ (see Fig. 6), bringing the flop count for the application of the WY transformations from $4kbn_b$ to $4k(b + n_b - 1)n_b$.

To overcome this overhead without losing the cache efficiency of BLAS-3 operations, we implemented our own optimized kernel routines [46]. We used cache blocking to exploit the memory hierarchy and loop unrolling to make use of vector operations (Dual FPU on BlueGene/P and SSE on x86). This means that in each step not all k eigenvectors are transformed at once, but only a subset of k_b eigenvectors. The number k_b should be set such that, first, the working set of $b \times k_b$ words fits into the (L1-) cache and, second, $k_b + 2$ words (a vector of length k_b and one element of the current Householder vector and eigenvector) fit into the registers to reduce the data transfer to the caches.

Optimized kernels were implemented for the PowerPC 450 (BlueGene/P) and for x86 processors. Figs. 5 and 7 compare the runtime of the WY and non-WY approach for the banded-to-tridiagonal back transformation of eigenvectors. The comparison of the different approaches (1D WY, 1D non-WY, 2D WY, 2D non-WY) reveals some platform and problem size dependence in detail, but the overall trends are clear. (i) The current 1D or 2D non-WY implementations almost always outperform the WY approaches, in particular for short Householder vectors (small b). (ii) The 2D approach is superior to 1D, in particular for the smaller Poly27069 problem and increasing numbers of cores. This can be understood because the 1D approach must stop scaling as the number of cores approaches the number of needed eigenvectors. The distribution of Householder vectors also becomes increasingly expensive in the 1D approach.

3. Partial eigensystems of symmetric tridiagonal matrices

The current release 1.8.0 of the ScaLAPACK library [33] provides three different methods for computing eigenvalues and eigenvectors of a symmetric tridiagonal matrix: `xSTEQQR2`, the implicit QL/QR method [47]; `PxSTEBZ` and `PxSTEIN`, a combination of bisection and inverse iteration (B&I) [48,49]; and `PxSTEDC`, the divide-and-conquer (D&C) method [50–52]. LAPACK 3.2.2 [31] and release 3.2 of the PLAPACK library [53] also provide the new MRRR algorithm [54,12], which will be included in a future ScaLAPACK release as well [55].

B&I is able to compute a subset of k eigenpairs at reduced cost, but for clustered eigenvalues this method may take $O(k^2n)$ operations or/and lose orthogonality. The MRRR algorithm can also compute subsets (at cost $O(kn)$), but the current implementations in rare cases fail to attain satisfying orthogonality; recent research towards improving its robustness is described in [56,57]. Compared with these two methods, QR/QL and D&C produce very good eigensystems, but both are designed to compute either all or no eigenvectors. In the “all eigenvectors” case, D&C typically is *much* faster than QR/QL, at the cost of $2n^2$ additional workspace. The same is true for D&C vs. B&I, as is well documented in the literature [12,58]. In practice D&C is therefore often used to compute all eigenvectors, even if only a subset of them is needed, and we restrict ourselves to this case here for space reasons.

In the following we will briefly review the D&C algorithm and show how it can be modified to compute a subset of the eigenvectors at reduced cost.

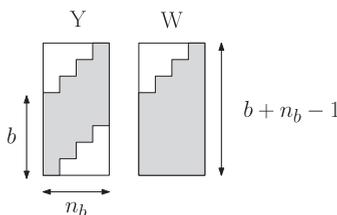


Fig. 6. Nonzero structure of the matrices W and Y during the back transformation of eigenvectors.

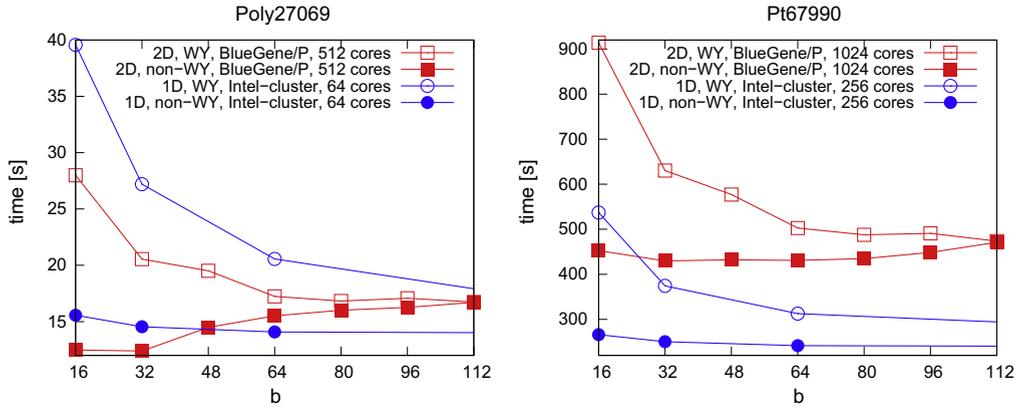


Fig. 7. Runtime of the tridiagonal-to-banded back transformation for Poly27069 and Pt67990 using different intermediate bandwidths b .

3.1. Outline of the divide-and-conquer algorithm

To determine the eigendecomposition $\mathbf{T} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ of a symmetric tridiagonal matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$, D&C proceeds as follows. **Split** the tridiagonal matrix into two half-sized submatrices:

$$\mathbf{T} = \begin{pmatrix} \mathbf{T}_1 & \\ & \mathbf{T}_2 \end{pmatrix} + \rho \mathbf{w} \mathbf{w}^T, \quad (3)$$

where \mathbf{T}_1 and \mathbf{T}_2 are of size $n_1 \approx n/2$ and $n_2 = n - n_1$, resp., and $\mathbf{w} \in \mathbb{R}^n$ has nonzeros only at positions n_1 and $n_1 + 1$.

Solve subproblems, i.e., call the D&C routine recursively to compute the eigendecompositions $\mathbf{T}_1 = \mathbf{Q}_1 \mathbf{\Lambda}_1 \mathbf{Q}_1^T$ and $\mathbf{T}_2 = \mathbf{Q}_2 \mathbf{\Lambda}_2 \mathbf{Q}_2^T$. If the subproblems are “small enough” then QR/QL may be used instead, which terminates the recursion. Note that by (3) we have

$$\mathbf{T} = \mathbf{Q}^{\text{sub}} (\mathbf{\Lambda}^{\text{sub}} + \rho \mathbf{z} \mathbf{z}^T) \mathbf{Q}^{\text{sub}T}, \quad (4)$$

where $\mathbf{Q}^{\text{sub}} = \text{diag}(\mathbf{Q}_1, \mathbf{Q}_2)$, $\mathbf{\Lambda}^{\text{sub}} = \text{diag}(\mathbf{\Lambda}_1, \mathbf{\Lambda}_2)$, and $\mathbf{z} = \mathbf{Q}^{\text{sub}T} \mathbf{w}$. Thus the next steps are aimed at computing the eigendecomposition for a rank-one perturbed diagonal matrix, $\mathbf{\Lambda}^{\text{sub}} + \rho \mathbf{z} \mathbf{z}^T = \hat{\mathbf{Q}} \mathbf{\Lambda} \hat{\mathbf{Q}}^T$.

Deflate. If some components z_i of \mathbf{z} are (almost) zero then the corresponding $(\lambda_i^{\text{sub}}, \mathbf{q}_{\mathbf{T}_i}^{\text{sub}})$ are also (within small error bounds) eigenpairs of \mathbf{T} . Similarly, if two eigenvalues in $\mathbf{\Lambda}^{\text{sub}}$ are identical (or close) then a zero (or small entry) in \mathbf{z} can be generated by applying a sequence of plane rotations, thus again leading to the first case. Assume that d eigenpairs can be computed cheaply in this way and that these are permuted to the end, that is, $\mathbf{P}^T \mathbf{R}^T \mathbf{z} = (\mathbf{z}_n^T, \mathbf{0})^T$, where \mathbf{R}^T is the product of all rotations, \mathbf{P}^T is the permutation, and \mathbf{z}_n contains the components that cannot be made zero. Then (4) reduces to

$$\mathbf{T} = \mathbf{Q}^{\text{sub}} \mathbf{R} \mathbf{P} \begin{pmatrix} \mathbf{\Lambda}_n^{\text{sub}} + \rho \mathbf{z}_n \mathbf{z}_n^T & \\ & \mathbf{\Lambda}_d^{\text{sub}} \end{pmatrix} (\mathbf{Q}^{\text{sub}} \mathbf{R} \mathbf{P})^T, \quad (5)$$

that is, the size of the rank-one perturbed diagonal eigenproblem is reduced by d (“deflation”). Note that $\mathbf{\Lambda}_d^{\text{sub}} \in \mathbb{R}^{d \times d}$ contains the deflated eigenvalues λ_i^{sub} corresponding to a zero in $\mathbf{P}^T \mathbf{R}^T \mathbf{z}$, whereas $\mathbf{\Lambda}_n^{\text{sub}}$ contains the $n - d$ non-deflated eigenvalues.

Solve the secular equation. The eigenvalues λ_i of the matrix $\mathbf{\Lambda}_n^{\text{sub}} + \rho \mathbf{z}_n \mathbf{z}_n^T$ are given by the solutions of the *secular equation*

$$f(\lambda) = 1 + \rho \sum_{k=1}^{n-d} \frac{\zeta_k^2}{\ell_k - \lambda} = 0, \quad (6)$$

where ζ_k and ℓ_k denote the k th component of \mathbf{z}_n and the k th diagonal entry in $\mathbf{\Lambda}_n^{\text{sub}}$, respectively. In theory, the corresponding eigenvectors are given by diagonal scalings $\hat{\mathbf{q}}_{\mathbf{T}_i} = (\mathbf{\Lambda}_n^{\text{sub}} - \lambda_i \mathbf{I})^{-1} \mathbf{z}_n$, but they must be computed in another way to ensure orthogonality.

Propagate eigenvectors. According to (5), the eigenvectors of \mathbf{T} are given by

$$\mathbf{Q} = \mathbf{Q}^{\text{sub}} \mathbf{R} \mathbf{P} \hat{\mathbf{Q}} = \begin{pmatrix} \mathbf{Q}_1 & \\ & \mathbf{Q}_2 \end{pmatrix} \left[\mathbf{R} \mathbf{P} \begin{pmatrix} \hat{\mathbf{Q}}_n & \\ & \mathbf{I}_d \end{pmatrix} \right].$$

To improve the efficiency of this computation, the matrix in square brackets is further permuted as $\begin{pmatrix} \mathbf{V}_{11} & \mathbf{V}_{12} & \mathbf{0} & \mathbf{V}_{14} \\ \mathbf{0} & \mathbf{V}_{22} & \mathbf{V}_{23} & \mathbf{V}_{24} \end{pmatrix}$, where the first and third block columns contain the eigenvectors from \mathbf{T}_1 (from \mathbf{T}_2 , resp.) that were not involved in any

Table 2

Time (in seconds) for computing partial eigensystems with D&C for the matrices from the Poly27069 and Pt67990 problems.

	128-CPU Intel cluster		1024-CPU BlueGene/P	
	$n = 27,069$	$n = 67,990$	$n = 27,069$	$n = 67,990$
1 eigenvector	7.0	78.5	7.3	69.2
20% eigenvectors	7.6	81.1	7.7	70.0
40% eigenvectors	10.3	100.0	9.3	79.4
60% eigenvectors	13.5	138.6	10.9	102.1
80% eigenvectors	16.4	184.7	12.7	128.8
All eigenvectors	19.4	223.9	14.3	149.1

Table 3

Time (in seconds) for computing partial eigensystems with D&C on a BlueGene/P.

	512 CPUs	1024 CPUs	2048 CPUs
Poly27069 ($\approx 13\%$)	13.5	7.3	5.7
Pt67990 ($\approx 64\%$)	201.4	105.1	62.4

rotation, the fourth block column contains the deflated eigenvectors, which need no further processing, and the second block column comprises the rest. Then the update actually involves two matrix–matrix products

$$\mathbf{Q}_1 \cdot (\mathbf{V}_{11}, \mathbf{V}_{12}) \quad \text{and} \quad \mathbf{Q}_2 \cdot (\mathbf{V}_{22}, \mathbf{V}_{23}). \quad (7)$$

Two factors contribute to the mostly excellent performance of the D&C method: most arithmetic operations take place in heavily optimized matrix–matrix products, and deflation often leads to a significant reduction of the complexity as compared to the $O(n^3)$ worst case.

3.2. Computing a subset of the eigenvectors with D&C

The recursive calls of the D&C algorithm correspond to a binary call tree. Profiling reveals that the final two matrix–matrix products (7) at the root of the recursion tree can account for up to 70% of the total time, unless extreme deflation takes place. In that case the matrix products contribute only a few percent to the total time.

If only a subset of the eigenvectors is required then only these must be propagated in the two products. This can be accomplished in two ways. Either only the wanted vectors are included in $(\mathbf{V}_{11}, \mathbf{V}_{12})$ and $(\mathbf{V}_{22}, \mathbf{V}_{23})$, thus reducing the size of the matrix products, or unwanted vectors are set to zero before the multiplication. The latter approach reduces memory traffic, but it only works with zero-aware `xGEMM` implementations.

Note that splitting the matrix into irreducible blocks, as done in the LAPACK routine `xSTEDC` (but not in the ScaLAPACK implementation), requires an additional preprocessing step. First, we must determine which eigenpairs are required from each block (e.g., eigenpairs 10:50 of the whole matrix may correspond to eigenpairs 3:18 of the first block, eigenpairs 5:13 of the second, and eigenpairs 4:19 of the third block). This functionality is provided by the routine `xLARXI` from our new MRRR implementation [56,57]. Then the modified D&C algorithm is called for each irreducible block with the respective sub-subset.

Table 2 shows that this modification yields considerable savings. If the matrix is large enough and not too few eigenvectors are sought then the method also scales well up to large numbers of processors; see Table 3.

Since at least 30% of D&C's work is spent outside the two top-level matrix–matrix multiplications, at most threefold speed-up can be achieved this way. Thus it is natural to ask if subsets can also lead to savings at deeper recursion levels. Unfortunately this seems not to be the case because the secular equation (6) involves all non-deflated eigenvalues, and these are independent from the subset.

4. Conclusions

We have presented improvements for two stages in the computation of eigensystems of symmetric matrices.

For the banded-to-tridiagonal back transformation of the eigenvectors, which is necessary if a two-step tridiagonalization has been used, a dynamic 2D data layout reduces the communication volume (as compared to a 1D layout) without incurring significant load imbalance (as would be the case with a static 2D layout). In addition, explicit cache blocking can reduce the computational overhead for the blocked application of the Householder transformations, as compared to WY-based techniques. Combined, these two approaches can yield significant improvements for the parallel efficiency, in particular for large

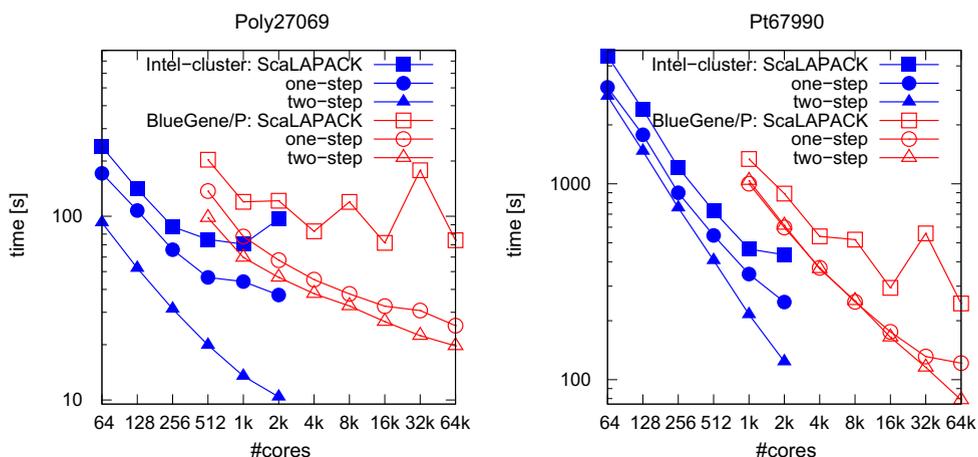


Fig. 8. Overall runtime of the symmetric eigenvalue problem. The ScaLAPACK implementation `pdsyevd` is compared to the implementation presented in [38] (one-step) and the two-step approach, presented in this paper. The one-step implementation as well as the two-step approach make use of the improved divide-and-conquer algorithm. For the tridiagonal-to-banded back transformation in each case the fastest implementation was used; see Fig. 5.

numbers of processor cores, as well as for the per-processor utilization, and they enable further large savings by using a two-step reduction to tridiagonal form.

The solution of the tridiagonal eigenproblem has been addressed, too. The popular divide-and-conquer tridiagonal eigensolver features high speed and accuracy, but it is designed to compute either all or no eigenvectors. We have presented a modification that allows to compute partial eigensystems at reduced cost, on serial and parallel machines.

Fig. 8 summarizes our findings and improvements. As can be seen, the vendor-optimized ScaLAPACK implementations (ESSL or MKL) always stop scaling, even for the very large Pt67990 problem. Our present two-step implementation is clearly more efficient than even our optimized one-step implementation [38] for both problems on the Intel cluster. Only on the BlueGene/P, and for the large problem Pt67990, does the one-step implementation remain competitive, due to the large ratio of needed eigenvectors to matrix size, $k/n \approx 64\%$. This indicates that the one-step tridiagonalization would be more beneficial if all eigenpairs were needed on BlueGene/P, as expected due to the higher operations count of the two-step approach, but remarkably, the same is not necessarily true on Intel/Infiniband.

For the measurements in Fig. 8 we used an intermediate bandwidth $b = 64$ on both machines. An optimal choice of b for the whole algorithm (reduction and back transformation) depends on the matrix size n , the number of desired eigenvectors k , the number of processes p , and last but not least on the used hardware. A larger b allows BLAS routines to operate near peak performance and decreases the number of messages sent, but it also increases the runtime of the reduction from banded to tridiagonal form. Experience has shown that $b \sim 50$ is a good choice; see also Fig. 7. The choice of an optimal parameter set will be the subject of further research.

While our research is driven by demands arising in electronic structure theory, the techniques described in this paper are general and applicable to any field where the solution of large symmetric eigenproblems is required.

Note added in proof

We also compared to the alternative tridiagonalization routine `pdsyntrd` in ScaLAPACK. On the BlueGene/P, `pdsyntrd` was significantly faster than `pdsytrd` for non-square processor grids (35:2 s vs. 81:8 s for the tridiagonalization of Poly27069 on 8192 processors with block size 16), while our one-step and two-step reductions remained superior (29:3 s and 21:6 s, resp.). The respective timings for 16384 processors were 43:5 s / 60:0 s / 26:2 s / 18:9 s. On the Intel cluster, `pdsyntrd` performed slightly worse than `pdsytrd`.

Acknowledgements

The authors want to thank the referees for their valuable comments that helped to improve the presentation and Jack Poulson for fruitful discussions on `pdsyntrd`.

References

- [1] W. Kohn, L. Sham, Self-consistent equations including exchange and correlation effects, *Phys. Rev.* 140 (1965) A1133–A1138.
- [2] A. Szabo, N. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*, Dover, 1996.
- [3] M. Rohlfing, S.G. Louie, Electron-hole excitations in semiconductors and insulators, *Phys. Rev. Lett.* 81 (1998) 2312–2315.

- [4] M. Casida, All-electron local and gradient-corrected density-functional calculations of Na_n dipole polarizabilities for $n = 1-6$, in: P. Chong (Ed.), *Recent Advances in Density Functional Methods, Part I*, World Scientific, Singapore, 1995, p. 155.
- [5] J.M. Soler, E. Artacho, J.D. Gale, A. Garcia, J. Junquera, P. Ordejon, D. Sanchez-Portal, The SIESTA method for ab initio order-N materials simulation, *J. Phys.: Condens. Matter* 14 (2002) 2745–2779.
- [6] M.J. Rayson, P.R. Briddon, Rapid iterative method for electronic-structure eigenproblems using localised basis functions, *Comput. Phys. Commun.* 178 (2008) 128–134.
- [7] M.J. Rayson, P.R. Briddon, Highly efficient method for Kohn–Sham density functional calculations of 500–10000 atom systems, *Phys. Rev. B* 80 (2009) 205104-1–205104-11.
- [8] V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter, M. Scheffler, Ab initio molecular simulations with numeric atom-centered orbitals, *Comput. Phys. Commun.* 180 (2009) 2175–2196.
- [9] P. Blaha, H. Hofstätter, O. Koch, R. Laskowski, K. Schwarz, Iterative diagonalization in augmented plane wave based methods in electronic structure calculations, *J. Comput. Phys.* 229 (2010) 453–460.
- [10] N.D.M. Hine, P.D. Hayes, A.A. Mostofi, M.C. Payne, Linear-scaling density-functional simulations of charged point defects in Al_2O_3 using hierarchical sparse matrix algebra, *J. Chem. Phys.* 133 (2010) 114111.
- [11] J. Vandevondele, J. Hutter, An efficient orbital transformation method for electronic structure calculations, *J. Chem. Phys.* 118 (2003) 4365–4369.
- [12] P. Bientinesi, I.S. Dhillon, R.A. van de Geijn, A parallel eigensolver for dense symmetric matrices based on multiple relatively robust representations, *SIAM J. Sci. Comput.* 27 (1) (2005) 43–66.
- [13] W.N. Gansterer, J. Zotti, Parallelization of divide-and-conquer eigenvector accumulation, in: J.C. Cunha, P.D. Medeiros (Eds.), *Euro-Par 2005*, Springer, Berlin, 2005, pp. 847–856.
- [14] Y. Zhou, Y. Saad, M.L. Tiago, J.R. Chelikowsky, Parallel self-consistent-field calculations via Chebyshev-filtered subspace acceleration, *Phys. Rev. E* 74 (2006) 066704-1–066704-8.
- [15] Y. Bai, R.C. Ward, Parallel block tridiagonalization of real symmetric matrices, *J. Parallel Distrib. Comput.* 68 (2008) 703–715.
- [16] V. Weber, J. VanDeVondele, J. Hutter, A. Niklasson, Direct energy functional minimization under orthogonality constraints, *J. Chem. Phys.* 128 (2008) 084113.
- [17] E. Polizzi, Density-matrix-based algorithm for solving eigenvalue problems, *Phys. Rev. B* 79 (2009) 115112-1–115112-6.
- [18] J.-L. Fattebert, Accelerated block preconditioned gradient method for large scale wave functions calculations in density functional theory, *J. Comput. Phys.* 229 (2010) 441–452.
- [19] M.J. Rayson, Rapid filtration algorithm to construct a minimal basis on the fly from a primitive Gaussian basis, *Comput. Phys. Commun.* 181 (2010) 1051–1056.
- [20] S. Goedecker, Linear scaling electronic structure methods, *Rev. Mod. Phys.* 71 (1999) 1085–1123.
- [21] D. Bowler, T. Miyazaki, M. Gillan, Recent progress in linear scaling ab initio electronic structure techniques, *J. Phys.: Condens. Matter* 14 (2002) 2781–2799.
- [22] C.-K. Skylaris, P.D. Haynes, A.A. Mostofi, M.C. Payne, Introducing ONETEP: linear-scaling density functional simulations on parallel computers, *J. Chem. Phys.* 122 (2005) 084119.
- [23] C.J. Garcia-Cervera, J. Lu, Y. Xuan, W. E, Linear-scaling subspace-iteration algorithm with optimally localized nonorthogonal wave functions for Kohn–Sham density functional theory, *Phys. Rev. B* 79 (2009) 115110-1–115110-13.
- [24] D. Bowler, T. Miyazaki, Calculations for millions of atoms with density functional theory: linear scaling shows its potential, *J. Phys.: Condens. Matter* 22 (2010) 074207.
- [25] E.R. Davidson, The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices, *J. Comput. Phys.* 17 (1975) 87–94.
- [26] A.V. Knyazev, Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method, *SIAM J. Sci. Comput.* 23 (2001) 517–541.
- [27] F. Bottin, S. Leroux, A. Knyazev, G. Zerah, Large-scale ab initio calculations based on three levels of parallelization, *Comput. Mater. Sci.* 42 (2008) 329–336.
- [28] D.M. Wood, A. Zunger, A new method for diagonalising large matrices, *J. Phys. A: Math. Gen.* 18 (1985) 1343–1359.
- [29] C. Vömel, S.Z. Tomov, O.A. Marques, A. Canning, L.-W. Wang, J.J. Dongarra, State-of-the-art eigensolvers for electronic structure calculations of large-scale nano-systems, *J. Comput. Phys.* 227 (2008) 7113–7124.
- [30] J. Iwata, D. Takahashi, A. Oshiyama, T. Boku, K. Shiraishi, S. Okada, K. Yabana, A massively-parallel electronic-structure calculations based on real-space density functional theory, *J. Comput. Phys.* 229 (2010) 2339–2363.
- [31] E. Anderson, Z. Bai, C.H. Bischof, L.S. Blackford, J.W. Demmel, J.J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D.C. Sorensen, *LAPACK Users' Guide*, third ed., SIAM, Philadelphia, PA, 1999.
- [32] V. Havu, V. Blum, P. Havu, M. Scheffler, Efficient $O(N)$ integration for all-electron electronic structure calculation using numeric basis functions, *J. Comput. Phys.* 228 (2009) 8367–8379.
- [33] L.S. Blackford, J.Y. Choi, A.J. Cleary, E. D'Azevedo, J.W. Demmel, I.S. Dhillon, J.J. Dongarra, S. Hammarling, G. Henry, A.P. Petitet, K.S. Stanley, D.W. Walker, R.C. Whaley, *ScaLAPACK Users' Guide*, SIAM, Philadelphia, PA, 1997.
- [34] C.H. Bischof, C.F. Van Loan, The WY representation for products of Householder matrices, *SIAM J. Sci. Stat. Comput.* 8 (1) (1987) s2–s13.
- [35] R.S. Schreiber, C.F. Van Loan, A storage-efficient WY representation for products of Householder transformations, *SIAM J. Sci. Stat. Comput.* 10 (1989) 53–57.
- [36] A.G. Sunderland, Parallel diagonalization performance on high-performance computers, in: R. Ciegis et al. (Eds.), *Parallel Scientific Computing and Optimization: Advances and Applications*, Springer, Berlin, 2009, pp. 57–66.
- [37] C. Bischof, B. Lang, X. Sun, Parallel tridiagonalization through two-step band reduction, in: *Proceedings of the Scalable High-Performance Computing Conference*, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 23–27.
- [38] R. Johanni, V. Blum, V. Havu, H. Lederer, M. Scheffler, (2011), in preparation.
- [39] P. Havu, V. Blum, V. Havu, P. Rinke, M. Scheffler, Large-scale surface reconstruction energetics of Pt(100) and Au(100) by all-electron density functional theory, *Phys. Rev. B* 82 (2010) 161418(R).
- [40] B. Lang, *Effiziente Orthogonaltransformationen bei der Eigen- und Singulärwertzerlegung*, Habilitationsschrift, Bergische Universität GH Wuppertal, Fachbereich Mathematik, 1997.
- [41] B. Lang, A parallel algorithm for reducing symmetric banded matrices to tridiagonal form, *SIAM J. Sci. Comput.* 14 (6) (1993) 1320–1338.
- [42] H.R. Schwarz, Tridiagonalization of a symmetric band matrix, *Numer. Math.* 12 (1968) 231–241.
- [43] C.H. Bischof, B. Lang, X. Sun, A framework for symmetric band reduction, *ACM Trans. Math. Softw.* 26 (4) (2000) 581–601.
- [44] C.H. Bischof, B. Lang, X. Sun, Algorithm 807: The SBR toolbox—software for successive band reduction, *ACM Trans. Math. Softw.* 26 (4) (2000) 602–616.
- [45] T. Auckenthaler, H.-J. Bungartz, T. Huckle, L. Krämer, B. Lang, P.R. Willems, Developing algorithms and software for the parallel solution of the symmetric eigenvalue problem, in press.
- [46] R. Wittmann, Development and implementation of purpose-built high performance linear algebra kernels for the use on supercomputers, Bachelor's thesis, Institut für Informatik, Technische Universität München, 2010.
- [47] J.G.F. Francis, The QR transformation: A unitary analogue to the LR transformation, Paart I and II, *Computer J.* 4 (1961/62) 265–274. 332–345.
- [48] J.W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.
- [49] J.W. Demmel, K.S. Stanley, The performance of finding eigenvalues and eigenvectors of dense symmetric matrices on distributed memory computers, in: *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, Philadelphia, PA, 1994, pp. 528–533.

- [50] J.J.M. Cuppen, A divide and conquer method for the symmetric tridiagonal eigenproblem, *Numer. Math.* 36 (1981) 177–195.
- [51] M. Gu, S.C. Eisenstat, A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, *SIAM J. Matrix Anal. Appl.* 16 (1) (1995) 172–191.
- [52] F. Tisseur, J.J. Dongarra, A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures, *SIAM J. Sci. Comput.* 20 (6) (1999) 2223–2236.
- [53] R.A. van de Geijn, *Using LAPACK*, MIT Press, Cambridge, MA, 1997.
- [54] I.S. Dhillon, A new $O(n^2)$ algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem, Ph.D. Thesis, University of California at Berkeley, 1997.
- [55] C. Vömel, ScaLAPACK's MRRR algorithm, *ACM Trans. Math. Softw.* 37 (1) (2009) 1–35.
- [56] P.R. Willems, On MRRR-type algorithms for the tridiagonal symmetric eigenproblem and the bidiagonal SVD, Ph.D. Thesis, Bergische Universität Wuppertal, Fachbereich Mathematik und Naturwissenschaften, 2010.
- [57] P.R. Willems, B. Lang, A framework for the MR³ algorithm: Theory and implementation, submitted for publication.
- [58] J.W. Demmel, O.A. Marques, B.N. Parlett, C. Voemel, Performance and accuracy of LAPACK's symmetric tridiagonal eigensolvers, *SIAM J. Sci. Comput.* 30 (2008) 1508–1526.