Japan Journal of Industrial and Applied Mathematics (2019) 36:699–717
https://doi.org/10.1007/s13160-019-00360-8

SPECIAL FEATURE: ORIGINAL PAPER
International Workshop on Eigenvalue Problems: Algorithms; Software and Applications, in Petascale Computing (EPASA2018)

Benefits from using mixed precision computations in the ELPA-AEO and ESSEX-II eigensolver projects

Andreas Alvermann, et al. ***[full author details at the end of the article]***

Received: 30 May 2018 / Revised: 12 January 2019 / Published online: 27 April 2019

Abstract
We first briefly report on the status and recent achievements of the ELPA-AEO (Eigen value Solvers for Petaflop Applications—Algorithmic Extensions and Optimizations) and ESSEX II (Equipping Sparse Solvers for Exascale) projects. In both collaboratory efforts, scientists from the application areas, mathematicians, and computer scientists work together to develop and make available efficient highly parallel methods for the solution of eigenvalue problems. Then we focus on a topic addressed in both projects, the use of mixed precision computations to enhance efficiency. We give a more detailed description of our approaches for benefiting from either lower or higher precision in three selected contexts and of the results thus obtained.

Keywords ELPA-AEO · ESSEX · Eigensolver · Parallel · Mixed precision

Mathematics Subject Classification 65F15 · 65F25 · 65Y05 · 65Y99

1 Introduction

Eigenvalue computations are at the core of simulations in various application areas, including quantum physics and electronic structure computations. Being able to best utilize the capabilities of current and emerging high-end computing systems is essential for further improving such simulations with respect to space/time resolution or by including additional effects in the models. Given these needs, the ELPA-AEO and ESSEX-II projects contribute to the development and implementation of efficient highly parallel methods for eigenvalue problems, in different contexts.

This work has been supported by the Deutsche Forschungsgemeinschaft through the priority programme 1648 "Software for Exascale Computing" (SPPEXA) under the project ESSEX-II and by the Federal Ministry of Education and Research through the project "Eigenvalue soLvers for Petaflop Applications—Algorithmic Extensions and Optimizations" (ELPA-AEO) under Grant No. 01H15001.

Extended author information available on the last page of the article

Both projects are aimed at adding new features (concerning, e.g., performance and resilience) to previously developed methods and at providing additional functionality with new methods. Building on the results of the first ESSEX funding phase [14,34], ESSEX-II again focuses on iterative methods for very large eigenproblems arising, e.g., in quantum physics. ELPA-AEO's main application area is electronic structure computation, and for these moderately sized eigenproblems direct methods are often superior. Such methods are available in the widely used ELPA library [19], which had originated in an earlier project [2] and is being improved further and extended with ELPA-AEO.

In Sects. 2 and 3 we briefly report on the current state and on recent achievement in the two projects, with a focus on aspects that may be of particular interest to prospective users of the software or the underlying methods. In Sect. 4 we turn to computations involving different precisions. Looking at three examples from the two projects we describe how lower or higher precision is used to reduce the computing time.

2 The ELPA-AEO project

In the ELPA-AEO project, chemists, mathematicians and computer scientists from the Max Planck Computing and Data Facility in Garching, the Fritz Haber Institute of the Max Planck Society in Berlin, the Technical University of Munich, and the University of Wuppertal collaborate to provide highly scalable methods for solving *moderately-sized* ($n \lesssim 10^6$) Hermitian eigenvalue problems. Such problems arise, e.g., in electronic structure computations, and during the earlier ELPA project, efficient direct solvers for them had been developed and implemented in the ELPA library [19].

This library is widely used (see https://elpa.mpcdf.mpg.de/about for a description and pointers to the software), and it has been maintained and further improved continually since the first release in 2011. The ELPA library contains optimized routines for the steps in the direct solution of generalized Hermitian positive eigenproblems $AX = BX\Lambda$, that is, (i) the Cholesky decomposition $B = U^H U$, (ii) the transformation $A \mapsto \widetilde{A} = U^{-H} A U^{-1}$ to a standard eigenproblem $\widetilde{A}\widetilde{X} = \widetilde{X}\Lambda$, (iii) the reduction of $\widetilde{A}$ to tridiagonal form, either in one step or via an intermediate banded matrix, (iv) a divide-and-conquer tridiagonal eigensolver, and (v) back-transformations for the eigenvectors corresponding to steps (iii) and (ii). A typical application scenario from electronic structure computations ("SCF cycle") requires a sequence of a few dozens of eigenproblems $A^{(k)}X = BX\Lambda$ to be solved, where the matrix B remains unchanged; see Sect. 4.3 for more details. ELPA is particularly efficient in this situation by explicitly building U^{-1} for steps (ii) and (v).

ELPA-AEO is aimed at further improving the performance of computations that are already covered by ELPA routines and at providing new functionality. In the remainder of this section we highlight a few recent achievements that may be of particular interest to current and prospective users of the library.

An alternative approach for the transformation (ii) has been developed [18], which is based on Cannon's algorithm [5]. The transformation is done with two matrix products: *multiplication 1* computes the upper triangle M_u of $M := A \cdot U^{-1}$, then M_u is transposed to obtain the lower triangle M_l of $M^H = U^{-H} A$, and finally *multipli-*

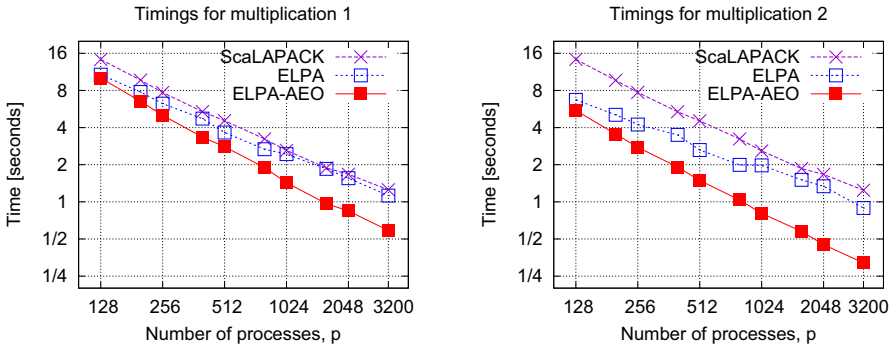

Fig. 1 Timings for the two multiplications in the transformation $A \mapsto \widetilde{A}$ with routines from ScaLAPACK, current ELPA routines, and the new implementations. The runs were made on the HYDRA system at MPCDF in Garching with 20 processes per node (two 10-core Intel Ivy bridge processors running at 2.8 GHz) and double precision real matrices of size $n = 30{,}000$. Process grids had aspect ratios 1 : 1 or 1 : 2; e.g., a 10×20 grid was set up for $p = 200$. With $p = 3200$, the new codes run at $\approx 40\%$ of the nodes' *peak* performance

```
autotune_handle = elpa_autotune_setup( handle, ELPA_AUTOTUNE_FAST,
                                       ELPA_AUTOTUNE_DOMAIN_REAL, &error ) ;
for ( i = 0 ; i < 20 ; i++ ) {
   unfinished = elpa_autotune_step( handle, autotuning_handle ) ;
   if ( unfinished == 0 )
      printf( "ELPA autotuning finished in the %d th SCF step\n", i ) ;

   /* Solve EV problem */
   elpa_eigenvectors( handle, a, ev, z, &error ) ;
}
elpa_autotune_best_set( handle, autotune_handle ) ;
elpa_autotune_deallocate( autotune_handle ) ;
```

Fig. 2 Using ELPA's autotuning facility to adjust the algorithmic parameters during the solution of (at most) twenty eigenvalue problems in an SCF cycle, and saving them for later use

cation 2 computes the lower triangle of $M_l \cdot U^{-1} = \widetilde{A}$. Both routines assume that one dimension of the process grid is a multiple of the other. They make use of the triangular structure of their arguments to save on computation and communication. The timing data in Fig. 1 show that the new implementations are highly competitive.

Recent ELPA releases provide extended facilities for performance tuning. The computational routines have an argument that can be used to guide the routines in selecting algorithmic paths (if there are different ways to proceed) and algorithmic parameters (such as block sizes) and to receive performance data from their execution. An easy-to-use autotuning facility allows setting such parameters in an automated way by screening the parameter space; see the code fragment in Fig. 2 for an example. Note that the parameter set obtained with the coarse probing induced by ELPA_AUTOTUNE_FAST might be improved later on.

In earlier releases, ELPA could be configured for single or double precision computations, but due to the naming conventions only one of the two versions could be linked to a calling program. Now, both precisions are accessible from one library, and mixing them may speed up some computations; see Sect. 4.3 for an example.

New functionality for addressing banded generalized eigenvalue problems will be added. An efficient algorithm for the transformation to a banded standard eigenvalue problem has been developed [17], and its parallelization is currently under way. This will complement the functions for solving banded standard eigenvalue problems that are already included in ELPA.

3 The ESSEX-II project

The ESSEX-II project is a collaborative effort of physicists, mathematicians and computer scientists from the Universities of Erlangen-Nuremberg, Greifswald, Tokyo, Tsukuba, and Wuppertal and from the German Aerospace Center in Cologne. It is aimed at developing exascale-enabled solvers for selected types of *very large* ($n \gg 10^6$) eigenproblems arising, e.g., in quantum physics; see the project's homepage at https://blogs.fau.de/essex/ for more information, including pointers to publications and software.

ESSEX-II builds on results from the first ESSEX funding phase, in particular the Exascale enabled Sparse Solver Repository (ESSR), which provides a (block) Jacobi–Davidson method, the BEAST subspace iteration-based framework, and the Kernel Polynomial Method (KPM) and Chebyshev time propagation for determining few extremal eigenvalues, a bunch of interior eigenvalues, and information about the whole spectrum and dynamic properties, respectively. The BEAST framework uses subspace iteration with Rayleigh–Ritz extraction of approximate eigenpairs. It provides three different basic methods for constructing the subspace and heuristic strategies for running them; more details will be given in Sect. 4.1.

Based on the versatile SELL-C-σ format for sparse matrices [13], the General, Hybrid, and Optimized Sparse Toolkit (GHOST) [15] contains optimized kernels for often-used operations such as sparse matrix times (multiple) vector products (optionally fused with other computations) and operations with block vectors, as well as a task manager, for CPUs, Intel Xeon Phi MICs and Nvidia GPUs and combinations of these. The Pipelined Hybrid-parallel Iterative Solver Toolkit (PHIST) [34] provides the eigensolver algorithms with interfaces to GHOST and other "computational cores," together with higher-level functionality, such as orthogonalization and linear solvers.

With ESSEX-II, the interoperability of these ESSR components will be further improved to yield a mature library, which will also have an extended range of applicability, including non-Hermitian and nonlinear eigenproblems. Again we highlight only a few recent achievements.

The Scalable Matrix Collection (ScaMaC) provides routines that simplify the generation of test matrices. The matrices can be chosen from several physical models, e.g., boson or fermion chains, and parameters allow adjusting sizes and physically motivated properties of the matrices. With 32 processes, a distributed size 2.36G matrix for a Hubbard model with 18 sites and 9 fermions can be set up in less than 10 minutes.

The block Jacobi–Davidson solver has been extended to non-Hermitian and generalized eigenproblems. It can be run with arbitrary preconditioners, e.g., the AMG preconditioner ML [31], and employs a robust and fast block orthogonalization scheme that can make use of higher-precision computations; see Sect. 4.2 for more details.

```
// BEAST init (omitted)

Checkpoint beast_checkpoint( "BEAST", comm ) ;
beast_checkpoint->add( "eigenvectors", &X ) ;
beast_checkpoint->add( "eigenvalues", &e ) ;
... // Some more
beast_checkpoint->add( "control_variables", &state ) ;
beast_checkpoint->commit() ;

beast_checkpoint->restartIfNeeded( NULL ) ;

// BEAST iterations
while ( !state.abort_condition ) {
   // Compute projector, etc. (omitted)
   ...
   beast_checkpoint->update() ;
   beast_checkpoint->write() ;
}
```

Fig. 3 Using the CRAFT library to checkpoint the current eigenvector approximations X and other quantities in every iteration of the main loop

The BEAST framework has been extended to seamlessly integrate three different approaches for spectral filtering in subspace iteration methods (polynomial filters, rational filters based on plain contour integration, and a moment-based technique) and to make use of their respective advantages with adaptive strategies. The BEAST framework also benefits from using different precisions; see Sect. 4.1.

At various places, measures for improving resilience have been included, based on verifying known properties of computed quantities and on checksums, combined with checkpoint–restart. To simplify incorporating the latter into numerical algorithms, the Checkpoint–Restart and Automatic Fault Tolerance (CRAFT) library has been developed [30]. Figure 3 illustrates its use within the BEAST framework. CRAFT can handle the GHOST and PHIST data types, as well as user-defined types. Checkpoints may be nested to accommodate, e.g., low-frequency high-volume together with high-frequency low-volume checkpointing in multilevel numerical algorithms, and the checkpoints can be written asynchronously to reduce overhead. By relying on the Scalable Checkpoint/Restart (SCR) and User-Level Failure Mitigation (ULFM-) MPI libraries, CRAFT also provides support for fast node-level checkpointing and for handling node failures.

4 Benefits of using a different precision

Doing computations in lower precision is attractive from a performance point of view because it reduces memory traffic in memory-bound code and, in compute-bound situations, allows more operations per second, due to vector instructions manipulating more elements at a time. However, the desired accuracy often cannot be reached in single precision and then only a part of the computations can be done in lower precision, or a correction is needed; cf., e.g., [3] for the latter. In Sect. 4.1 we describe an approach for reducing overall runtimes of the BEAST framework by using lower-precision computations for early iterations.

Higher precision, on the other hand, is often a means to improve robustness. It is less known that higher precision can also be beneficial w.r.t. runtime. This is demonstrated in Sect. 4.2 in the context of orthogonalization.

In Sect. 4.3 we come back to using lower precision, from the perspective of an important application area: self-consistent field (SCF) cycles in electronic structure computations. Each iteration of such a cycle requires the solution of a generalized eigenproblem (GEP). After briefly introducing the context, we discuss how ELPA-AEO's features can be used to steer the precision from the application code, targeting either the entire solution of a GEP or particular steps within its solution.

4.1 Changing precision in subspace iteration-based eigensolvers

The BEAST framework [9,34] is aimed at finding those eigenpairs (λ, x) of a generalized interior eigenproblem $Ax = Bx\lambda$ (A Hermitian, B Hermitian positive definite) with λ in a given search interval $I_\lambda = [\underline{\lambda}, \overline{\lambda}]$, in particular for interior eigenvalues. It is based on subspace iteration with spectral filtering and Rayleigh–Ritz extraction, that is, a subspace U containing an approximate basis for the desired eigenvectors is constructed from some initial vectors Y, then a Rayleigh–Ritz step is used to obtain the approximate eigenpairs. If the desired residual threshold is not yet reached, we iterate, using the approximate eigenvectors in our choice of Y for the following iteration; cf. also Fig. 5 below. The main distinguishing factor of the variants BEAST-P/-C/-M in our framework is the construction of the subspace U.

BEAST-P, which is only applicable for standard eigenproblems, implements a polynomial filter [22,26], using matrix–(block) vector products to apply a polynomial in A to Y,

$$U = \sum_{j=0}^{N} \omega_j A^j Y.$$

In both BEAST-C and BEAST-M, the filter is applied via quadrature approximations of contour integrals of the form

$$r(B^{-1}A) \approx \frac{1}{2\pi i} \int_\Gamma z^k (zB - A)^{-1} B \, \mathrm{d}z,$$

where Γ is a contour in the complex plane enclosing the sought eigenvalues and no others. BEAST-C follows Polizzi's FEAST algorithm [23] in computing

$$U = \sum_{j=1}^{N} w_j (z_j B - A)^{-1} BY$$

with suitable nodes z_j and weights w_j. This requires N linear solves for each iteration of the eigensolver, with an $n \times m$ block vector of right hand sides Y. BEAST-M realizes a specific Sakurai–Sugiura method [27], Sakurai–Sugiura Rayleigh–Ritz [28]. Here,

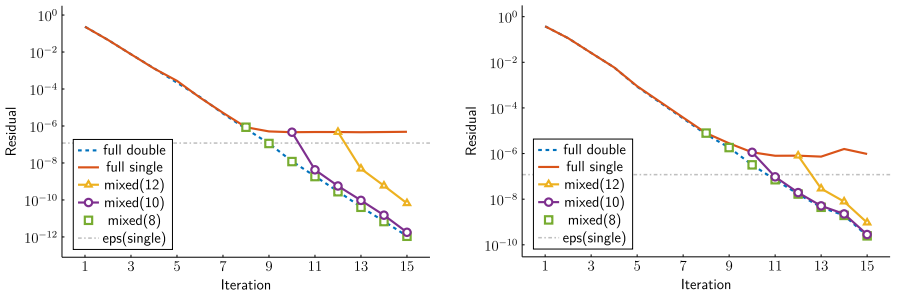

Fig. 4 Smallest residual $\min_i \|Ax_i - \lambda_i x_i\|$ (left picture) and geometric mean $(\prod_i \|Ax_i - \lambda_i x_i\|)^{1/k}$ of the residuals (right picture) over BEAST-P iterations (with polynomial degree 50) for computations done completely in double precision, completely in single precision, and with switching from single to double precision after iterations 12, 10, and 8. The horizontal line indicates the single precision machine epsilon. The minimum and mean, resp., are taken over those k approximate eigenpairs that are classified as lying within the search interval

the subspace is constructed as

$$U = \left[U_0, \ldots, U_{s-1}\right], \quad \text{where} \quad U_k = \sum_{j=1}^{N} w_j z_j^k (z_j B - A)^{-1} BY.$$

Thus, again N linear solves must be performed as in BEAST-C, but since the overall subspace is computed as a combination of their solution, Y needs only $(1/s)$th the desired number of columns of U, which can reduce the cost of the linear solves. It should be noted that a traditional Sakurai–Sugiura Rayleigh–Ritz implementation requires very few, or only one iteration, with a large overall subspace size. However, we consider it here within the context of a constrained subspace size, making it a truly iterative method.

We first consider the effect of starting with single precision and switching to double precision in later iterations. Since BEAST is designed to behave iteratively, we expect that this effect should be limited. Figure 4 shows BEAST-P's progress (smallest and average residual of the current approximations in each iteration) in solving a standard eigenproblem $AX = X\Lambda$ for a size 3200 topological insulator matrix A from the ESSEX repository [1] and $I_\lambda = [-0.5, 0.5]$, which contains 36 eigenpairs. We see that the residuals for single precision data and computations are very close to those obtained with double precision, until we reach the single precision barrier. Continuing in single precision leads to stagnation. By contrast, if we switch to double precision data and computations sufficiently before the barrier, convergence proceeds as if the entire run was in double precision. Even a later switch need not have dramatic effects; we see that convergence, although stalled temporarily by the single precision barrier, proceeds at the same rate and possibly even slightly faster when switched two and four iterations "too late." In the case of 10 iterations in single precision (two past the ideal of 8), the overall residual reached after 15 total iterations is again close to that of the full double and ideal switch computations.

Choose desired subspace size m (> number of evals in I_λ) and initial vectors Y
while *not converged* **do**
  Construct subspace $U \leftarrow Y$ with BEAST-* scheme
  Resize subspace based on rank(U)
  Solve reduced eigenproblem $A_U W = B_U W \Lambda$, where $A_U = U^* A U$, $B_U = U^* B U$
  $X := UW$
  $Y := BX$ (BEAST-P/-C) or $Y := BXR$ (BEAST-M, with a random matrix R)
  If single precision barrier has been reached, switch to double precision
end

Fig. 5 The mixed-precision BEAST framework. Computations are started in single precision and may be continued in double precision

A switching strategy based on this observation is shown in Fig. 5. In Fig. 6 we report results for using this approach to solve the problem $AX = \Lambda X$ for a size 16M graphene matrix from the ESSEX repository and $I_\lambda = [-0.0025, 0.0025]$. The computation was done on the Emmy cluster at the University of Erlangen-Nuremberg, using 32 nodes, each with two Xeon 2660v2 chips. All methods computed an identical number of 318 eigenpairs in I_λ to a tolerance of 10^{-10}. BEAST-P exhibits a remarkable similarity in convergence rates between single and double precision before the switch threshold, and the mixed precision run was roughly 1.2 times faster than using double precision throughout. In BEAST-C the rates are again similar; due to a few unconverged eigenpairs, the double precision computation required an additional iteration of the eigensolver for this problem, enabling a higher speedup 1.4 for the mixed precision version. In BEAST-M, we observe some stagnation before the switch threshold, and an additional iteration was required in the mixed precision run. In this case, the mixed precision run was slower than pure double precision, with a "speedup" of 0.9. Overall, the reduction in time from early iterations performed in single precision shows most clearly for BEAST-P. We note that the actual speed-up observed between single and double precision depends on both the hardware and software used; higher optimization of vectorized instructions or the use of accelerators such as GPUs could produce a more dramatic time difference.

The results indicate that initial iterations in single precision may have a limited effect on the overall convergence of the eigensolver if an appropriate switching point to double precision is chosen, thus allowing for a reduction in cost without sacrificing accuracy. We plan to combine this approach with relaxed stopping criteria for solving the linear systems in BEAST-C and -M iteratively; cf. also [9,10] for related work.

4.2 Using higher precision for robust and fast orthogonalization

In contrast to the standard Jacobi–Davidson method, which determines the sought eigenpairs one-by-one, the block Jacobi–Davison method in ESSEX [24] computes them by groups. Here we will consider only the real standard eigenvalue problem $Av_i = v_i\lambda_i$. Then one iteration contains the following two major steps:

1. Given n_b current approximations $\tilde{\lambda}_i$ and $\tilde{v}_i$, $i = 1, \ldots, n_b$, and a set of previously converged Schur vectors $W = (w_1, \ldots, w_k)$ $(k \geq 0)$, use some steps of a (blocked)

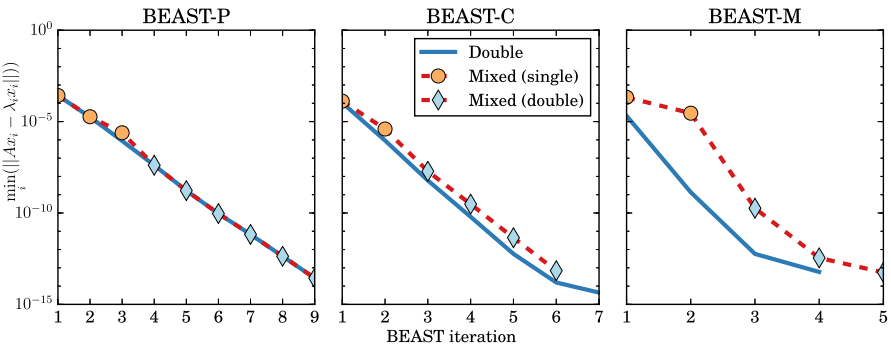

Fig. 6 Smallest residual over BEAST iterations for runs done completely in double precision (solid lines) and for mixed precision runs (dashed lines; different markers for iterations done in single and double precision, respectively), with algorithmic parameters set as follows. Size of U: 1.5× the estimated number of eigenvalues in the interval; polynomial degree of BEAST-P: 10,000; 4 Gauss-Legendre integration points for BEAST-C and 8 for BEAST-M, which is more sensitive to a low number of nodes; STRUMPACK direct solver [25] for the linear systems in BEAST-C and -M; threshold for the switch from single to double precision: 10^{-5} for BEAST-P and -C, and 10^{-4} for BEAST-M to prevent excessive stagnation

iterative linear solver for the correction equation

$$(I - \tilde{W}\tilde{W}^T)(A - \tilde{\lambda}_i I)(I - \tilde{W}\tilde{W}^T)x_i \;=\; -r_i, \quad i = 1, \ldots, n_b,$$

where $r_i = A\tilde{v}_i - \tilde{v}_i\tilde{\lambda}_i$ are the current residuals and $\tilde{W} = (W \mid \tilde{v}_1, \ldots, \tilde{v}_{n_b})$.

2. Obtain new directions $y_1, \ldots, y_{n_b}$ by orthogonalizing the x_i against W and among themselves (the y_i are then used to update the $\tilde{v}_i$).

The block method typically requires more operations than the non-blocked one and therefore has previously not been advocated, but in [24] it has been shown that this drawback can be more than outweighed by allowing the use of kernels that can be implemented to make best use of the capabilities of modern processors (in particular, sparse matrix times multiple vectors), such that the block method tends to run faster. In addition, it is more robust in the presence of multiple or tightly clustered eigenvalues.

In the following we focus on the orthogonalization in step 2. It is well known that if one first orthogonalizes the x_i against W ("*phase I*") and then among themselves ("*phase II*"), the second phase can spoil the results of the first one; this also holds if we reverse the order of the phases. By contrast, a robust algorithm is obtained by iterating this process, alternating between the two phases and using a rank-revealing technique in phase II; see [12,33] for a thorough discussion.

We follow this approach, using a plain projection $\tilde{Y} = (I - WW^T)X$ for phase I and SVQB [32] on $\tilde{Y}$ for phase II. We prefer SVQB over TSQR [8] because the bulk of computation may be done in a highly performant matrix–matrix multiplication for building the Gram matrix $M = \tilde{Y}^T\tilde{Y}$. This would also be true for CholQR [32], but SVQB is superior in the following sense (in practice, however, the advantage is subtle and hard to show in experiments as we are iterating anyway).

Both methods orthogonalize $\tilde{Y}$ by determining a suitable matrix $Z \in \mathbb{R}^{n_b \times n_b}$ such that $Z^T M Z = I$, and setting $Y = \tilde{Y}Z$; this yields $Y^T Y = I$. For SVQB we take $Z =$

$U\Lambda^{-1/2}$, where $M = U\Lambda U^T$ is an eigendecomposition of M, whereas a (possibly pivoted, partial) Cholesky decomposition $M = R^T R$ is used for setting $Z = R^{-1}$ in CholQR. A sufficient condition for minimizing the amplification of rounding errors in the final multiplication $\tilde{Y}Z$, is that Z should be as close as possible to the identity matrix. So we have to solve the optimization problem

$$\min_{Z \in \mathbb{R}^{n_b \times n_b},\, Z^T M Z = I} \|Z - I\|.$$

For the Frobenius norm, this is a special case of the orthogonal Procrustes problem analyzed by Schönemann in [29], as it can be transformed to the following formulation:

$$\min_{\hat{Z} \in \mathbb{R}^{n_b \times n_b},\, \hat{Z}^T \hat{Z} = I} \|M^{-1/2}\hat{Z} - I\|_F.$$

As shown in [29], a solution can be constructed as $\hat{Z} = UU^T$ with the eigendecomposition $M^{1/2} = UD^{1/2}U^T$ (in [29] a more general case is considered exploiting a singular value decomposition). So the choice $Z = M^{-1/2}\hat{Z} = UD^{-1/2}$, that is, the SVQB algorithm, is optimal in the sense discussed above (for simplicity of the presentation we have assumed M to be full-rank, thus symmetric positive definite. The argumentation also can be extended to the rank-deficient case). This optimality argument does not mean that CholQR cannot be competitive in practice, in particular if it is done twice; see [35] for an error analysis of that method.

Our aim is to obtain a robust and fast overall orthogonalization method with fewer iterations by using extended precision computations; cf. [36,37] for related ideas in the context of CholQR and communication-avoiding GMRES.

In contrast to [36,37], we use extended precision throughout the orthogonalization, including the orthogonalization against W and the computation and decomposition of the Gram matrix. Our own kernels are based on the techniques described in [20] for working with numbers represented by two doubles (DD). Some of the kernels take standard double precision data and return DD results, others also take DD data as inputs. They make use of AVX2, Intel's *advanced vector extensions*, with FMA (fused multiply–add) operations; see [20, Chapter 5]. As proposed there, divisions and square roots are computed using the Newton–Raphson method. It should be noted that DD does not preclude the possibility of overflow or underflow in the construction of the Gramian. We did not take special precautions for this situation.

Figure 7 shows the results of a single two-phase orthogonalization, without iteration, for synthetic test matrices with varying condition. If X is ill-conditioned then TSQR does a much better job on X than SVQB, but this does not carry over to orthogonality against W, and using DD kernels can improve both orthogonalities by at least two orders of magnitude.

On modern architectures, even the performance of matrix–matrix multiplications such as $\tilde{V}^T\tilde{V}$ is memory-bound if the matrix $\tilde{V} \in \mathbb{R}^{n \times n_b}$ is only very few columns wide. Then the additional arithmetic operations required in the DD kernels come almost for free, and operations on small $n_b \times n_b$ matrices are cost negligible, even in extended precision. Figure 8 compares timings for the overall orthogonalization with a

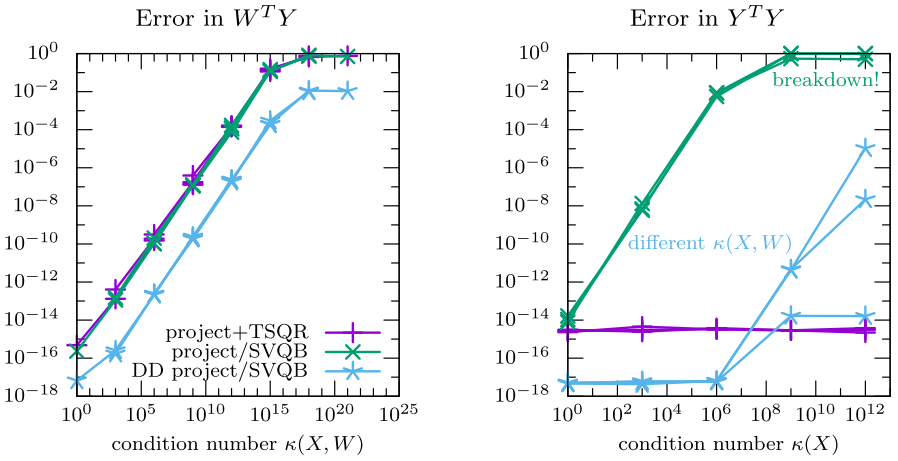

Fig. 7 Accuracy after one iteration (phase I and phase II) for synthetic test matrices $W \in \mathbb{R}^{n\times k}$, $X \in \mathbb{R}^{n\times n_b}$, where $n = 1000$, $k = 20$, and $n_b = 4$

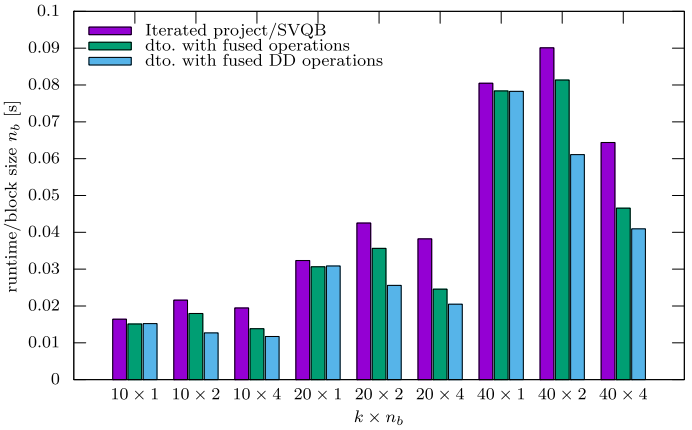

Fig. 8 Runtime "per vector" for overall orthogonalization of $X \in \mathbb{R}^{n\times n_b}$ against $W \in \mathbb{R}^{n\times k}$ and X (phases I and II, iterated until convergence at $\epsilon = 10^{-10}$) on an Intel Haswell workstation. $\kappa(X) = 10^{-6}$, $\kappa(X, W) = 10^{-12}$, $n = 8 \cdot 10^6$, k and n_b are indicated on the horizontal axis

straight-forward implementation, one that uses kernel fusion (combining several basic operations to further reduce memory accesses; not discussed here), and one with fused DD kernels. It reveals that using DD routines can even reduce overall time because the higher accuracy achieved in each iteration can lead to a lower number of iterations to reach convergence.

This technique can be useful for any algorithm that requires orthogonalizing a set X of vectors with respect to themselves and to another set W of (already orthonormal) vectors. It also extends to B-inner products, which is important, e.g., when solving generalized eigenvalue problems.

4.3 Mixed precision in SCF cycles with ELPA-AEO

The solution of the quantum-mechanical electronic-structure problem is at the basis of studies in computational chemistry, solid state physics, and materials science. In density-functional theory (DFT), the most wide-spread electronic-structure formalism, this implies finding the electronic density $n(\mathbf{r})$ that minimizes ($E_0 = \min E[n(\mathbf{r})]$) the convex total-energy functional $E[n(\mathbf{r})]$ under the constraint that the number of electrons, $N = \int \mathrm{d}\mathbf{r}\, n(\mathbf{r})$, is conserved. Here, the set of $3M$ nuclear coordinates $\{\mathbf{R}\}$ enters $E[n(\mathbf{r})]$ parametrically. Formally, this variational problem requires to find the stationary solution of the eigenvalue problem (EVP)

$$H[n(\mathbf{r})]\,\Psi(\mathbf{r}) = \varepsilon\Psi(\mathbf{r}) \quad \text{with} \quad n(\mathbf{r}) = \sum_{s=1}^{N} |\Psi_s(\mathbf{r})|^2 \tag{1}$$

in Hilbert space by iteratively updating $n(\mathbf{r})$, which depends on the N eigenstates Ψ_s with the lowest eigenvalues ε_s. This so called self-consistent field (SCF) cycle runs until "self-consistency" is achieved, i.e., until the mean interaction field contained in $H[n_k(\mathbf{r})]$ and/or other quantities (see below) do not change substantially between iterations anymore. In each step of the SCF cycle, the integro-differential equation (1) has to be solved. In practice, this is done by algebraizing Eq. (1) via a basis set expansion $\Psi_s = \sum_i x_{si}\varphi_i(\mathbf{r})$ of the so called orbitals in terms of appropriately chosen basis functions $\varphi_i(\mathbf{r})$, e.g., plane waves, localized functions, etc. By this means, one obtains a generalized EVP

$$A[n(\mathbf{r})]\,x = \lambda B x\ ,$$

in which the Hamiltonian A and the overlap matrix B are defined as

$$A_{ij}[n(\mathbf{r})] = \int \mathrm{d}\mathbf{r}\,\varphi_i^*(\mathbf{r})\,H[n(\mathbf{r})]\,\varphi_j(\mathbf{r}) \quad \text{and} \quad B_{ij} = \int \mathrm{d}\mathbf{r}\,\varphi_i^*(\mathbf{r})\,\varphi_j(\mathbf{r})\ . \tag{2}$$

As becomes clear from Eq. (2), the size of the EVP is thus determined by the number K of basis functions $\varphi_i(\mathbf{r})$ employed in the calculation. For efficient, atom-centered basis functions the ratio N/K of required eigenstates to matrix dimension typically ranges between 10 and 50%, rendering a direct solver competitive.

One SCF cycle yields the total energy $E_0(\{\mathbf{R}\})$ for just one set of nuclear coordinates $\{\mathbf{R}\}$. Studying molecules and materials requires the exploration of the high dimensional potential-energy surface (PES) which is given by $E_0(\{\mathbf{R}\})$ as a function of $\{\mathbf{R}\}$, e.g., via molecular dynamics (MD), statistical (e.g. Monte Carlo) sampling, or minimization and saddle point search algorithms. Accordingly, a typical computational study requires thousands if not millions of SCF cycles (about 10–100 SCF steps per cycle) to be performed in a single simulation. This large number of SCF steps makes it mandatory to investigate strategies to reduce the computational effort. Since only the final result of each converged SCF cycle is of physical relevance at all, the SCF procedure can be accelerated by using single precision (SP) routines instead of

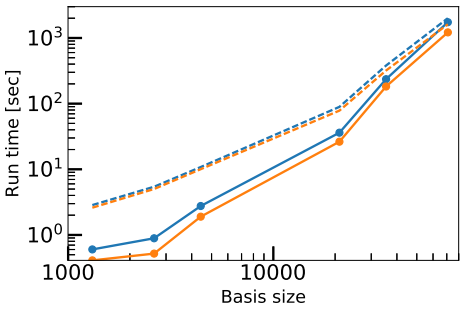

Fig. 9 Total time for one SCF cycle (dashed lines) and total time spent in ELPA-AEO (solid lines) as function of the basis set size using SP (orange) and DP routines (blue) for zirconia (ZrO_2, M from 6 to 768 atoms, N from 112 to 14,336 electrons). The calculations were performed with 8 Intel Xeon E5-2698v3 CPUs (4 nodes, 32 cores/node) and 4 GB of RAM per core

double precision (DP) ones in the appropriate eigensolver steps (cf. Sect. 2), as long as the final converged result is not altered up to the precision mandated by the problem at hand. The eigensolver steps discussed in this section are the Cholesky decomposition (i), the transformation to the standard eigenproblem (ii), and its standard diagonalization, which combines tridiagonalization (iii) and the tridiagonal eigensolver (iv), as defined in Sect. 2.

To showcase the importance of the readily available SP routines in ELPA-AEO, we have performed DFT calculations with the all-electron, numeric atomic orbitals based code FHI-aims [4], which supports both ELPA and ELPA-AEO through the ELSI package [38]. For this purpose, we have run benchmark calculations for zirconia (ZrO_2) in its tetragonal polymorph, a wide band-gap insulator often employed as thermal insulator in aeronautic applications [6,7]. Supercells containing between $M = 6$ and 768 atoms ($N = 112$ and 14,336 electrons) were investigated using the PBEsol exchange-correlation functional, "light" defaults for the numerical settings, and chemical species-specific "Tier 1" defaults for the basis functions φ_i. Accordingly, this translates to basis sets yielding matrix dimensions from $K = 1312$ to 70,848 for the investigated systems. The finite **k**-point grid required to sample reciprocal space to model such extended materials using periodic boundary conditions was chosen in such a way that the **k**-point density is roughly constant (between 128 and 216 **k**-points in the respective primitive Brillouin zone). As an example, Fig. 9 shows the total time for one SCF step and the total time spent in solving the EVP with SP and DP as function of the system size. Here, SP is only used in the diagonalization [steps (iii) and (iv) introduced in Sect. 2]. For larger system sizes (more than 10^4 basis functions), the computational time spent in the calculation of $A[n(\mathbf{r})]$, which typically exhibits linear scaling with respect to N in FHI-aims [11], becomes increasingly negligible compared to the EVP, which starts to dominate the computational time due to its cubic scaling. Switching from DP to SP thus allows for computational savings in the solution of the EVP on the order of 30–50%. Even for medium system sizes ($M = 96$ with $K = 2624$ basis functions) that are routinely addressed in DFT calculations [7] this already translates into savings in total computational time of around 10%, while savings of more than 20% are observed for larger systems (up to over 40% in Fig. 12).

However, SP routines cannot be exploited during the full SCF cycle: once a certain accuracy is reached, further SP SCF iterations do no longer approach con-

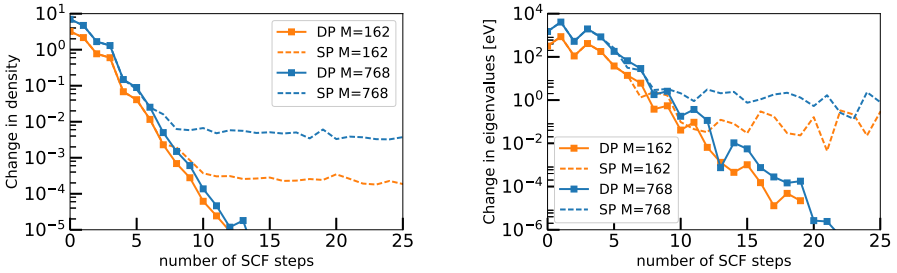

Fig. 10 Change in density (left) and eigenvalues (right) as function of the number of SCF steps during a full SCF cycle when single-(squares) or double-precision (triangles) routines are used. Calculations were performed for zirconia ($M = 162$ and $M = 768$ atoms, $N = 3024$ and 14,336 electrons, respectively, using the settings of Fig. 9)

vergence. This is demonstrated for ZrO_2 in Fig. 10. In each SCF step, we monitor two properties that are typically used for determining the convergence of such calculations: (I) the change in charge density between two subsequent steps k and $k+1$, $\Delta n = \int \mathrm{d}\mathbf{r}\, |n_k(\mathbf{r}) - n_{k+1}(\mathbf{r})|$, and (II) the change in the sum of the N lowest eigenvalues, $\Delta\varepsilon = \sum_{s=1}^{N} \varepsilon_s^{(k)} - \varepsilon_s^{(k+1)}$. For $M = 162$ atoms, we observe that Δn stalls at approximately $2.5 \cdot 10^{-4}$ electrons after the 10th SCF iteration for the calculation using SP. Similarly, $\Delta\varepsilon$ stalls at a value of $5 \cdot 10^{-2}$ eV, showing a less regular behavior, both in SP and DP. This can be traced back to the fact that the total-energy functional is not variational with respect to the eigenvalues. As also shown in Fig. 10 for $M = 768$ atoms ($N = 14{,}336$ electrons), the observed thresholds at which using SP no longer guarantees approaching convergence is, however, system and size dependent, since the respective quantities (energy, density, sum of eigenvalues, etc.) are extensive with system size, i.e., they scale linearly with the number of electrons, N. For these reasons, convergence criteria in DFT calculations are typically not chosen with respect to extensive quantities as the total energy, but with respect to intensive quantities, such as the total energy per atom. Hence the fraction of iterations for which SP routines can be used ($> 30\%$) are roughly independent of the system size, given that both the target quantity and its change, e.g., $n(\mathbf{r})$ and $\Delta n(\mathbf{r})$, are extensive with system size.

In general, not only the central steps (iii) and (iv) of solving the EVP (the diagonalization comprising the reduction to tridiagonal form and the tridiagonal eigensolver; cf. Sect. 2), but also the Cholesky decomposition (i) and the transformation to the standard eigenproblem (ii) offer the flexibility to choose between SP and DP. Even though the overlap-matrix B remains constant during an SCF cycle for an atom-centered basis, the test calculations on an AB-stacked graphite system ($M = 108$ atoms, PBE exchange-correlation functional, "tight" numerical defaults, "Tier 2" default basis set, $K = 23{,}682$ basis functions, $N = 648$ electrons) include the Cholesky decomposition in every iteration step in order to assess the impact of SP versus DP in step (i). Figure 11 illustrates that SP in (i) and (ii) does not noticeably change the convergence behavior of the extensive properties (change of total energy $\Delta E[n(\mathbf{r})] = E[n^{(k)}(\mathbf{r})] - E[n^{(k+1)}(\mathbf{r})]$, eigenvalues $\Delta\varepsilon$, and density Δn) during one SCF cycle and hence, full convergence is achieved in contrast to SP in the diagonal-

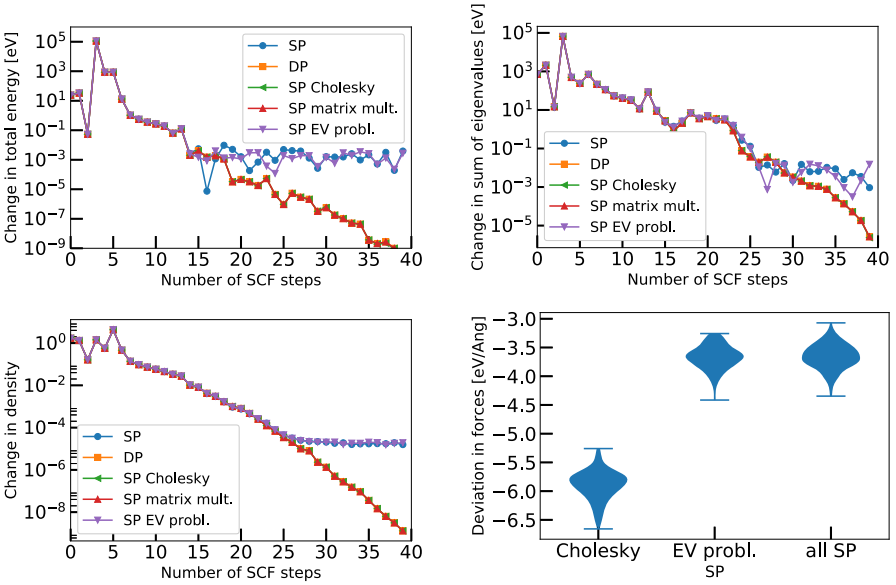

Fig. 11 Convergence behavior of change in total energy (top left), change in sum of eigenvalues (top right), and change in density (bottom left) with the number of SCF iterations for AB-stacked graphite ($M = 108$ atoms, $N = 648$ electrons, $K = 23{,}682$). SP in the Cholesky decomposition (step (i), green triangles) or the matrix multiplication (step (ii), red triangles) show convergence identical to full DP (orange squares) calculations and are essentially indistinguishable. SP in the diagonalization [steps (iii) and (iv), violet triangles] corrupts the convergence behavior. A combination of SP in steps (i) through (iv) (blue dots) does not reach convergence either. On the bottom right, the deviations in the forces per atom between SP and DP calculations are depicted for SP in different steps of the solution of the EVP

ization (iii) and (iv). This is confirmed in the bottom right picture in Fig. 11, where the forces on each atom, i.e., the gradients $\mathbf{F}_I = -\nabla_{\mathbf{R}_I} E_0(\{\mathbf{R}\})$ and their deviation from the full double precision values $|\mathbf{F}_I^{DP} - \mathbf{F}_I^{SP}|$ are shown. The force per atom, an intensive quantity, is typically monitored and required to reach a certain accuracy in calculations targeted at exploring $E_0(\{\mathbf{R}\})$. The bottom right plot in Fig. 11 confirms that SP in the Cholesky decomposition (i) influences the results only marginally; SP transformation (ii) even yields numerically identical results (not shown on the logarithmic scale). By contrast, a SP diagonalization results in force deviations of up to 0.5 meV/Å, which will still be sufficiently small for certain applications such as prescreening in PES exploration or statistical methods based on sampling by MD, when interpreting the error noise in the forces as acceptable thermal noise [16]. For the combination of SP throughout steps (i) to (iv), the convergence behavior and the force deviations are dominated by the performance of the eigensolver steps (iii) and (iv), and the convergence criteria for neither energy, eigenvalues, nor density are fulfilled. However, as discussed for Figs. 9 and 10, resorting to a diagonalization (iii) and (iv) in SP during the initial SCF steps is computationally advantageous, but switching to DP is required in the final steps for full convergence.

Figure 12 shows that the discussed advantages of SP are preserved in massively parallelized computations. Here, we display calculations for a slab of silicon carbide,

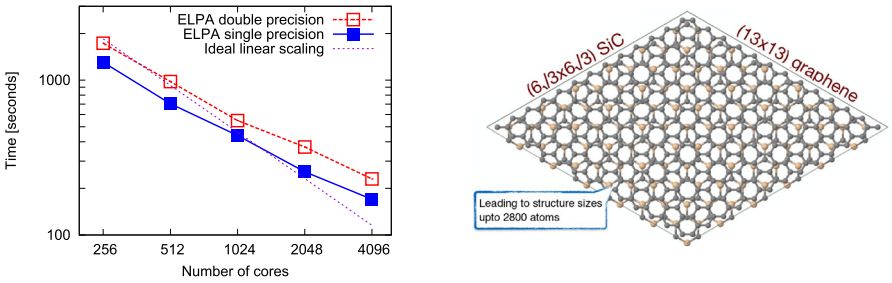

Fig. 12 Time for solving the EVP for Zero-Layer Graphene ($M = 1742$, 65,346 basis functions, LDA, Γ-point only) as function of the number of cores with DP (empty squares) and SP (filled squares). The calculations were performed with the IBM iDataPlex HPC system HYDRA using 2.6 GHz Intel Sandy Bridge EP nodes with 16 cores per node

where a layer of graphene is adsorbed on the surface [21]. Compared to the 2013 ELPA code base, which presents a common usage scenario before the ELPA-AEO project, we observe a speed-up of 1.7 for DP calculations. Another factor of 1.4 is obtained when switching to SP, which would not have been possible with earlier releases of the library. The almost ideal strong scaling with respect to the number of cores is retained in SP calculations.

5 Concluding remarks

The ESSEX-II and ELPA-AEO projects are collaborative research efforts targeted at developing iterative solvers for very large scale eigenproblems (dimensions $\gg$ 1M) and direct solvers for smaller-scale eigenproblems (dimensions up to 1M), and at providing software for these methods. After briefly highlighting some recent progress in the two projects w.r.t. auto-tuning facilities, resilience, and added functionality, we have discussed several ways of using mixed precision for reducing the runtime.

In iterative schemes such as BEAST, single precision may be used in early iterations. This need not compromise the final accuracy if we switch to double precision at the right time. Even working in extended precision may speed up the execution if the extra precision leads to fewer iterations and is not too expensive, as seen with an iterative orthogonalization scheme for the block Jacobi–Davison method. Additional finer-grained control of the working precision, addressing just particular steps of the computations can also be beneficial; this has been demonstrated with electronic structure computations, where the precision for each step was chosen directly from the calling code.

Our results indicate that the users should be able to adapt the working precision, as well as algorithmic parameters, to their particular needs, together with heuristics for automatic selection. Work towards these goals will be continued in both projects.

Acknowledgements The authors thank the unknown referees for their valuable comments that helped to improve and clarify the presentation.

References

1. Alvermann, A., Basermann, A., Fehske, H., Galgon, M., Hager, G., Kreutzer, M., Krämer, L., Lang, B., Pieper, A., Röhrig-Zöllner, M., Shahzad, F., Thies, J., Wellein, G.: ESSEX: Equipping sparse solvers for exascale. In: Lopes, L., et al. (eds.) Euro-Par 2014: Parallel Processing Workshops, LNCS, Springer, vol. 8806, pp. 577–588 (2014)
2. Auckenthaler, T., Blum, V., Bungartz, H.J., Huckle, T., Johanni, R., Krämer, L., Lang, B., Lederer, H., Willems, P.R.: Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations. Parallel Comput. **37**(12), 783–794 (2011)
3. Baboulin, M., Buttari, A., Dongarra, J., Kurzak, J., Langou, J., Langou, J., Luszczek, P., Tomov, S.: Accelerating scientific computations with mixed precision algorithms. Comput. Phys. Comm. **180**(12), 2526–2533 (2009)
4. Blum, V., Gehrke, R., Hanke, F., Havu, P., Havu, V., Ren, X., Reuter, K., Scheffler, M.: Ab initio molecular simulations with numeric atom-centered orbitals. Comput. Phys. Comm. **180**, 2175–2196 (2009)
5. Cannon, L.E.: A cellular computer to implement the Kalman filter algorithm. Ph.D. thesis, Montana State University, Bozeman, MT (1969)
6. Carbogno, C., Levi, C.G., Van de Walle, C.G., Scheffler, M.: Ferroelastic switching of doped zirconia: modeling and understanding from first principles. Phys. Rev. B **90**, 144109 (2014)
7. Carbogno, C., Ramprasad, R., Scheffler, M.: Ab Initio Green–Kubo approach for the thermal conductivity of solids. Phys. Rev. Lett. **118**(17), 175901 (2017)
8. Demmel, J., Grigori, L., Hoemmen, M., Langou, J.: Communication-optimal parallel and sequential QR and LU factorizations. SIAM J. Sci. Comput. **34**(1), A206–A239 (2012)
9. Galgon, M., Krämer, L., Lang, B.: Improving projection-based eigensolvers via adaptive techniques. Numer. Linear Algebra Appl. **25**(1), e2124 (2017)
10. Gavin, B., Polizzi, E.: Krylov eigenvalue strategy using the FEAST algorithm with inexact system solves. Numer. Linear Algebra Appl. p. e2188 (2018)
11. Havu, V., Blum, V., Havu, P., Scheffler, M.: Efficient $O(N)$ integration for all-electron electronic structure calculation using numeric basis functions. J. Comput. Phys. **228**(22), 8367–8379 (2009)
12. Hoemmen, M.: Communication-avoiding Krylov subspace methods. Ph.D. thesis, University of California, Berkeley (2010)
13. Kreutzer, M., Hager, G., Wellein, G., Fehske, H., Bishop, A.R.: A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units. SIAM J. Sci. Comput. **36**(5), C401–C423 (2014)
14. Kreutzer, M., Thies, J., Pieper, A., Alvermann, A., Galgon, M., Röhrig-Zöllner, M., Shahzad, F., Basermann, A., Bishop, A.R., Fehske, H., Hager, G., Lang, B., Wellein, G.: Performance engineering and energy efficiency of building blocks for large, sparse eigenvalue computations on heterogeneous supercomputers. In: Bungartz, H.J., Neumann, P., Nagel, W.E. (eds.) Software for Exascale Computing—SPPEXA 2013–2015, LNCSE, vol. 113, pp. 317–338. Springer, Switzerland (2016)
15. Kreutzer, M., Thies, J., Röhrig-Zöllner, M., Pieper, A., Shahzad, F., Galgon, M., Basermann, A., Fehske, H., Hager, G., Wellein, G.: GHOST: Building blocks for high performance sparse linear algebra on heterogeneous systems. Int. J. Parallel Prog. **45**(5), 1046–1072 (2016)
16. Kühne, T.D., Krack, M., Mohamed, F.R., Parrinello, M.: Efficient and accurate Car-Parrinello-like approach to Born-Oppenheimer molecular dynamics. Phys. Rev. Lett. **98**(6), 066401 (2007)
17. Lang, B.: Efficient reduction of banded hermitian positive definite generalized eigenvalue problems to banded standard eigenvalue problems. SIAM J. Sci. Comput. **41**(1), C52–C72 (2019)
18. Manin, V., Lang, B.: Cannon-type triangular matrix multiplication for the reduction of generalized hpd eigenproblems to standard form (2018) **(Submitted)**
19. Marek, A., Blum, V., Johanni, R., Havu, V., Lang, B., Auckenthaler, T., Heinecke, A., Bungartz, H.J., Lederer, H.: The ELPA library: Scalable parallel eigenvalue solutions for electronic structure theory and computational science. J. Phys.: Condens. Matter **26**(21), 213201 (2014)
20. Muller, J.M., Brisebarre, N., de Dinechin, F., Jeannerod, C.P., Lefèvre, V., Melquiond, G., Revol, N., Stehlé, D., Torres, S.: Handbook of Floating-Point Arithmetic. Springer, Berlin (2010)
21. Nemec, L., Blum, V., Rinke, P., Scheffler, M.: Thermodynamic equilibrium conditions of graphene films on SiC. Phys. Rev. Lett. **111**(6), 065502 (2013)

22. Pieper, A., Kreutzer, M., Alvermann, A., Galgon, M., Fehske, H., Hager, G., Lang, B., Wellein, G.: High-performance implementation of Chebyshev filter diagonalization for interior eigenvalue computations. J. Comput. Phys. **325**, 226–243 (2016)
23. Polizzi, E.: Density-matrix-based algorithm for solving eigenvalue problems. Phys. Rev. B **79**(11), 115112 (2009)
24. Röhrig-Zöllner, M., Thies, J., Kreutzer, M., Alvermann, A., Pieper, A., Basermann, A., Hager, G., Wellein, G., Fehske, H.: Increasing the performance of the Jacobi–Davidson method by blocking. SIAM J. Sci. Comput. **37**(6), C697–C722 (2015)
25. Rouet, F.H., Li, X.S., Ghysels, P., Napov, A.: A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization. ACM Trans. Math. Softw. **42**(4), 27:1–27:35 (2016)
26. Saad, Y.: Numerical Methods for Large Eigenvalue Problems, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia (2011)
27. Sakurai, T., Sugiura, H.: A projection method for generalized eigenvalue problems using numerical integration. J. Comput. Appl. Math. **159**(1), 119–128 (2003)
28. Sakurai, T., Tadano, H.: CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems. Hokkaido Math. J. **36**, 745–757 (2007)
29. Schönemann, P.H.: A generalized solution of the orthogonal Procrustes problem. Psychometrika **31**(1), 1–10 (1966)
30. Shahzad, F., Thies, J., Kreutzer, M., Zeiser, T., Hager, G., Wellein, G.: CRAFT: A library for easier application-level checkpoint/restart and automatic fault tolerance (2017). Preprint: arXiv:1708.02030 **(Submitted)**
31. Song, W., Wubs, F., Thies, J., Baars, S.: Numerical bifurcation analysis of a 3D turing-type reaction-diffusion model. Commun. Nonlinear Sci. Numer. Simul. **60**, 145–164 (2018)
32. Stathopoulos, A., Wu, K.: A block orthogonalization procedure with constant synchronization requirements. SIAM J. Sci. Comput. **23**(6), 2165–2182 (2002)
33. Stewart, G.W.: Block Gram–Schmidt orthogonalization. SIAM J. Sci. Comput. **31**(1), 761–775 (2008)
34. Thies, J., Galgon, M., Shahzad, F., Alvermann, A., Kreutzer, M., Pieper, A., Röhrig-Zöllner, M., Basermann, A., Fehske, H., Hager, G., Lang, B., Wellein, G.: Towards an exascale enabled sparse solver repository. In: Bungartz, H.J., Neumann, P., Nagel, W.E. (eds.) Software for Exascale Computing—SPPEXA 2013–2015, LNCSE, vol. 113, pp. 295–316. Springer, Switzerland (2016)
35. Yamamoto, Y., Nakatsukasa, Y., Yanagisawa, Y., Fukaya, T.: Roundoff error analysis of the Cholesky QR2 algorithm. Electron. Trans. Numer. Anal. **44**, 306–326 (2015)
36. Yamazaki, I., Tomov, S., Dong, T., Dongarra, J.: Mixed-precision orthogonalization scheme and adaptive step size for improving the stability and performance of CA-GMRES on GPUs. In: Daydé, M.J., Marques, O., Nakajima, K. (eds.) High Performance Computing for Computational Science—VECPAR 2014—11th International Conference, Eugene, OR, USA, June 30–July 3, 2014, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 8969, pp. 17–30. Springer (2014)
37. Yamazaki, I., Tomov, S., Dongarra, J.: Mixed-precision Cholesky QR factorization and its case studies on multicore CPU with multiple GPUs. SIAM J. Sci. Comput. **37**(3), C307–C330 (2015)
38. Yu, V.W., Corsetti, F., García, A., Huhn, W.P., Jacquelin, M., Jia, W., Lange, B., Lin, L., Lu, J., Mi, W., Seifitokaldani, A., Vázquez-Mayagoitia, Á., Yang, C., Yang, H., Blum, V.: ELSI: A unified software interface for Kohn-Sham electronic structure solvers. Comput. Phys. Comm. **222**, 267–285 (2018)

Affiliations

Andreas Alvermann[1] · Achim Basermann[2] · Hans-Joachim Bungartz[3] · Christian Carbogno[4] · Dominik Ernst[5] · Holger Fehske[1] · Yasunori Futamura[6] · Martin Galgon[7] · Georg Hager[5] · Sarah Huber[7] · Thomas Huckle[3] · Akihiro Ida[8] · Akira Imakura[6] · Masatoshi Kawai[8] · Simone Köcher[9] · Moritz Kreutzer[5] · Pavel Kus[10] · Bruno Lang[7] ·

Hermann Lederer[10] · Valeriy Manin[7] · Andreas Marek[10] · Kengo Nakajima[8] · Lydia Nemec[9] · Karsten Reuter[9] · Michael Rippl[3] · Melven Röhrig-Zöllner[2] · Tetsuya Sakurai[6] · Matthias Scheffler[4] · Christoph Scheurer[9] · Faisal Shahzad[5] · Danilo Simoes Brambila[4] · Jonas Thies[2] · Gerhard Wellein[5]

✉ Bruno Lang
lang@math.uni-wuppertal.de

1 Institute of Physics, University of Greifswald, Greifswald, Germany

2 German Aerospace Center (DLR), Cologne, Germany

3 Department of Informatics, Technical University of Munich, Munich, Germany

4 Fritz Haber Institute of the Max Planck Society, Berlin, Germany

5 High Performance Computing, University of Erlangen-Nuremberg, Erlangen, Germany

6 Applied Mathematics, University of Tsukuba, Tsukuba, Japan

7 Mathematics and Natural Sciences, University of Wuppertal, Wuppertal, Germany

8 Computer Science, The University of Tokyo, Tokyo, Japan

9 Department of Theoretical Chemistry, Technical University of Munich, Munich, Germany

10 Max Planck Computing and Data Facility, Garching, Germany