An Independently Blocked Algorithm for the Symmetric Eigenproblem

Wilfried N. Gansterer
Institute for Applied and Numerical Mathematics
Vienna University of Technology
ganst@aurora.tuwien.ac.at

Dieter F. Kvasnicka
Institute for Physical and Theoretical Chemistry
Vienna University of Technology
kvasnicka@tuwien.ac.at

Karlheinz Schwarz
Institute for Physical and Theoretical Chemistry
Vienna University of Technology
kschwarz@email.tuwien.ac.at

Christoph W. Ueberhuber
Institute for Applied and Numerical Mathematics
Vienna University of Technology
christof@uranus.tuwien.ac.at

January 1999

AURORA TR1999-01

The work described in this report was supported by the Special Research Program SFB F011 "AURORA" of the Austrian Science Fund.

Abstract

Blocking the tridiagonalization part of symmetric eigensolvers enables a significant efficiency enhancement. Various steps of a two-sweep tridiagonalization can be blocked independently of each other. Thus, a great deal of parallelism becomes utilizable. Consequently, Level 3 BLAS can be applied even in the reduction phase. A successful application of blocked eigensolvers in material science calculations is presented.

Contents

Chapter 1

Problem Specification

Symmetric eigensolvers play an important role in many scientific applications (see, for example, Schwarz, Blaha [25]). General information about the numerical solution of eigenproblems is found, for example, in Golub, Van Loan [20], Parlett [24], or Wilkinson [27].

Allan and Bush [1] present an overview over parallel diagonalization routines and compare advantages and disadvantages of the implemented methods. The majority of routines is available for real dense symmetric matrices only, although also Hermitian or sparse matrices are considered. The algorithms are of one of three kinds.

- Reduction based methods, which perform a Householder reduction to tridiagonal form and then solve the tridiagonal eigenproblem, for example using QL or QR iteration, bisection and inverse iteration, or divide and conquer methods;
- One-sided or two-sided Jacobi methods (see, for example, Hegland [22]);
- Lanczos methods (Cullum, Willoughby [12]) and subspace decomposition (Auslander, Tsao [3]).

In terms of arithmetic complexity, i. e., the number of floating-point operations required, reduction based methods are usually most efficient for dense matrices. In a parallel environment, however, they show limited scalability. Therefore the following recommendations are made in Allan and Bush [1].

- For an $n \times n$ matrix on P processors, where $n/P < 8$, they achieved best results with a one-sided Jacobi method.
- For matrices with $n \leq 3000$ and $n/P > 8$, reduction based methods performed best.

- For matrices with $n > 3000$ and $n/P > 8$, they recommend a two-sided Jacobi method.

This report discusses techniques which make reduction based methods applicable for a wider range of parameters (i) by improving sequential execution time and (ii) by improving scalability.

The Symmetric Eigenproblem

This report concentrates on the efficient computation of a subset of the eigenpairs (eigenvalues and eigenvectors) of a dense symmetric (Hermitian) matrix $A \in \mathbb{R}^{n\times n}$ ($\mathbb{C}^{n\times n}$), i. e., the solution of the symmetric (Hermitian) eigenproblem $AX = X\Lambda$. The efficiency of the method described strongly depends on the number m of the eigenvectors computed. If only eigenvalues have to be computed, efficiency is very high. If only some eigenvectors have to be computed the method is still competitive. If all eigenvectors have to be computed, efficiency may decrease significantly, but the method presented may still be a good choice on certain computers (see Gansterer et al. [19]).

The method used has three fundamental stages.

- Reduce matrix A to a band matrix $B = Q_b^\top A Q_b$. Matrix B has typically a small bandwidth b (for example, $b = 8$).
- Solve the reduced eigenproblem $BX_b = X_b\Lambda$.
- Transform the eigenvector matrix X_b of the reduced problem back to the eigenvector matrix $X = Q_b X_b$ of the original problem.

The banded eigenproblem is solved in another three stages.

- Reduce $BX_b = X_b\Lambda$ to a tridiagonal eigenproblem $TX_t = X_t\Lambda$.
- Compute the eigenvalues of the tridiagonal eigenproblem using bisection.
- Using these eigenvalues, compute the eigenvectors X_b of matrix B using inverse iteration.

Compared to eigensolvers using a one-sweep tridiagonal reduction (for example, the routine `LAPACK/dsyevx`), which perform 50 % of their floating point operations in Level 2 Blas, the first band reduction of two-sweep algorithms can be implemented utilizing Level 3 Blas exclusively. High level Blas (i) are advantageous to exploit memory hierarchies on sequential processors, and (ii) lead to better scaling behavior in parallel implementations.

Chapter 2

Existing Work

If all eigenvalues are required, the solution of the tridiagonal eigenproblem often dominates execution time. The tridiagonal eigenproblem has been worked on extensively (see, for example, Dhillon [13] and references therein). Dongarra et al. [16] and Bischof et al. [6] have significantly contributed to improving the efficiency of the reduction step.

2.1 LAPACK

Lapack (Anderson et al. [2]) makes use of high level Blas (Dongarra et al. [14, 15]) to exploit features of modern computers. In particular, Level 3 Blas exploit memory hierarchies to the highest degree. This is why most Lapack routines are based on Level 3 Blas. Unfortunately, one-sweep tridiagonalization can utilize Level 3 Blas for only 50 % of all floating-point operations due to intricate data dependencies (see Dongarra et al. [16], Gansterer et al. [17]).

2.2 SBR Toolbox

Bischof et al. [5, 6, 8] developed algorithms for the band reduction of a matrix and implemented them in the Sbr (successive band reduction) toolbox (Bischof et al. [7]). They showed that it is possible to tridiagonalize a matrix in several sweeps in a blocked fashion, relying on Level 3 Blas for the majority of floating-point operations.

Using the Sbr toolbox they developed multi-sweep eigensolvers. However, if eigenvectors are to be computed from the tridiagonal matrix T, these multi-sweep methods cannot compete with eigensolvers utilizing one-sweep tridiagonalization.

2.3 Computation of Eigenvectors from an Intermediate Band Matrix

Back-transformation of eigenvectors is much faster if the eigenvectors are computed from the intermediate band matrix B and not from T (see Gansterer et al. [17, 18, 19] and Fig. 2.1). The band width b of B has to be small, though, to enable a fast inverse iteration. This, unfortunately, limits the block size of the first reduction sweep.

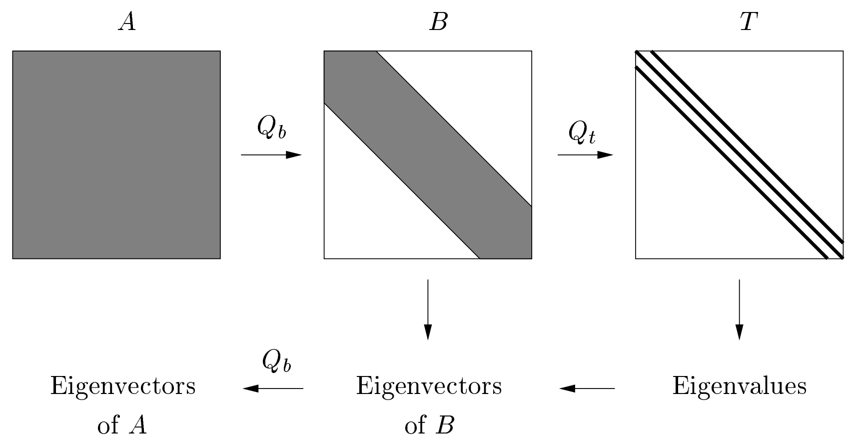

Figure 2.1: Structure of a two-sweep symmetric eigensolver.

2.4 Parallel Eigensolvers

Parallel eigensolvers can be found, for instance, in ScaLapack (Blackford et al. [9]), or in PLapack (van de Geijn [26]). Hendrickson et al. [23] have implemented a similar parallel eigensolver. One-sided Jacobi methods can be found, for example, in Hegland [22]. Parallel invariant subspace decomposition has been investigated in the Prism project (Auslander, Tsao [3], Bischof et al. [4]). The Sbr toolbox (see Section 2.2) was developed as part of Prism (Parallel Research on Invariant Subspace Methods). A prototype parallel implementation of the Sbr toolbox has been developed (Wu et al. [28]).

Chapter 3

Independently Blocked Eigensolver

Two-sweep tridiagonal reduction, utilizing the SBR toolbox (Bischof et al. [7]), is an efficient scheme if eigenvectors are to be computed from an intermediate band matrix B (see the performance curve denoted "two-sweep blocked" in Fig. 3.1 and Gansterer et al. [17, 18, 19]).

3.1 Improved Two-Sweep Reduction

To further improve the performance of two-sweep eigensolvers *independently blocked* algorithms have been developed (Wu et al. [28]). To explain the improved two-sweep technique, one-sweep reduction is discussed first.

One-Sweep Reduction

To reduce the symmetric matrix A to a symmetric tridiagonal matrix $T = Q^\top AQ$ orthogonal similarity transformations are used. Normally, Q is not generated explicitly. Instead of calculating $Q = H_2H_3 \dots H_{n-2}$ the Householder matrices $H_i = I - \tau u_i u_i^\top$ are applied to A iteratively for $i = 2, 3, \dots, n-1$:

$$A_i = H_i^\top A_{i-1} H_i \qquad \text{with} \qquad A_1 \equiv A, \;\; A_{n-1} \equiv T.$$

The application of each Householder matrix is done in three steps:

Step 1: $y_i = A_{i-1}u_i$,
Step 2: $v_i = y_i - \frac{1}{2}u_i(y_i^\top u_i)$,
Step 3: $A_i = A_{i-1} - u_i v_i^\top - v_i u_i^\top$.

In the routine `LAPACK/dsytrd` Step 3 is recast as a block operation by replacing k consecutive rank-2 updates by one rank-$2k$ update (see Section 2.1 and Dongarra et al. [16]). This restructuring enables the utilization of Level 3 BLAS which reuse data as much as possible and therefore maximize the ratio of floating-point operations to memory references. The resulting one-sweep algorithm still has an efficiency bottleneck. Step 1 involves a matrix-vector product which cannot be blocked. This Level 2 operation requires approximately 50 % of the floating-point operations of the tridiagonalization process, but it is responsible for up to 80 % of the execution time in a sequential solver (see Haunschmid, Kvasnicka [21]). This means, if the 80 % share were reduced to 50 %, a speedup by a factor of 2.5 could be achieved.

Two-Sweep Reduction

In *two-sweep* algorithms (see Section 2.2 and Bischof et al. [5, 6, 7]) all vectors occurring in the tridiagonalization phase can be replaced by matrix panels. Fig. 2.1 illustrates the sequence of operations in a two-sweep symmetric eigensolver. Two-sweep eigensolvers can be implemented using Level 3 BLAS in Step 3 *and* Step 1. However, the block size has to be chosen carefully. Large block sizes help to reduce the communication overhead on parallel machines. Moreover, they are often appropriate to match the memory hierarchy. On the other hand, the block size chosen for the first sweep, i. e., in the reduction of A to the band matrix B, is a lower bound for the band width of B. Large block sizes and thus large band widths require much extra work to be done in the second sweep and in the computation of the eigenvectors.

Due to these conflicting factors an optimum block size exists for each computer and for each problem size. The execution time can be even further reduced by introducing and optimizing two different block sizes instead of one single block size: b_1 in the matrix-matrix product corresponding to Step 1 and b_3 in the rank-$2k$ update corresponding to Step 3. Thus an *independently blocked* algorithm results. This algorithm subsumes one-sweep and two-sweep block algorithms: (i) one-sweep blocking (see Section 2.1) corresponds to setting $b_1 = 1$, and (ii) two-sweep blocking as mentioned in Section 2.2 corresponds to setting $b_1 = b_3$.

In principle, both the eigenvalues and the eigenvectors could be computed directly from the band matrix B. Execution time can be improved, however, if the eigenvalues are computed from the tridiagonal matrix. The eigenvectors are computed from the band matrix, because this helps to significantly speed up the back-transformation. The accuracy of the eigenvectors is not affected, provided good approximations of the respective eigenvalues are available.

The floating-point performance of various sequential eigensolvers—the independently blocked algorithm, the one-sweep algorithm, the two-sweep algorithm, and the unblocked algorithm—is illustrated in Fig. 3.1.

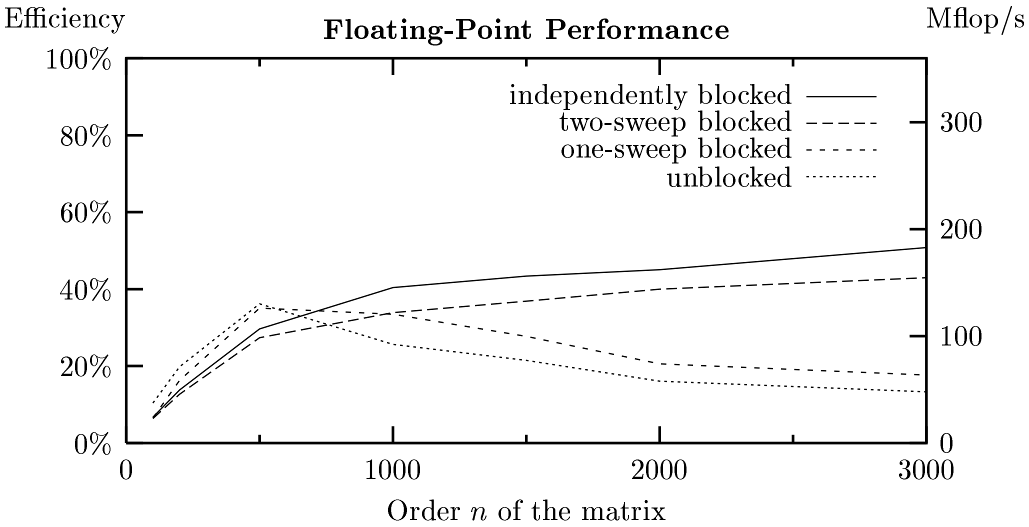

Figure 3.1: Performance of symmetric eigensolvers on one processor of an SGI Power Challenge. 10 % of the eigenvalues and eigenvectors are computed. Efficiency is given in terms of the theoretical peak performance of 360 Mflop/s.

The independently blocked band reduction algorithm (see Algorithm 1) makes it possible to choose b_3 large enough to achieve satisfactory floating-point performance in the rank-$2k$ update. Independently, b_1 can be chosen (i) to meet the requirements of the communication network and (ii) to keep the computational overhead of the second sweep and of the eigenvector computation small. As a result, the number of accesses to matrix A is reduced: $\lceil n/b_1 \rceil$ symmetric matrix multiplications are necessary to generate the update matrices in the inner loop and $\lceil n/b_3 \rceil$ symmetric rank-$2k$ updates are required to apply the update matrices in the outer loop, whereas the unblocked algorithm requires n symmetric matrix-vector products and n symmetric rank-2 updates.

3.2 Parallelization

Not all code sections of symmetric eigensolvers have to be parallelized. If only a few (or no) eigenvectors are to be computed, a parallelization of the

Algorithm 1

n ... order of the matrix A
p ... block size b_3 of the outer loop (for example, $p = 60$)
$\bar{p}$ ... block size b_1 of the inner loop (for example, $\bar{p} = 8$)

$N = (n-2)/p$
$\bar{N} = p/\bar{p}$
$A_0 = A$
do $k = 1, 2, \ldots, N$
    $s = (k-1)p + 1$
    do $j = 1, 2, \ldots, \bar{N}$
        $\bar{s} = (k-1)p + (j-1)\bar{p} + 1$
        do for $m \in \{\bar{s}, \bar{s}+1, \ldots, \bar{s}+\bar{p}-1\}$
            $a_m = a_m - \sum_{l=\bar{s}}^{m-1} (v_l^{(m)} u_l + u_l^{(m)} v_l)$
        end do
        find $(u_{\bar{s}}, u_{\bar{s}+1}, \ldots, u_{\bar{s}+\bar{p}+1})$
            by QR factorizing $(a_{\bar{s}}, a_{\bar{s}+1}, \ldots, a_{\bar{s}+\bar{p}+1})$
        do for $m \in \{\bar{s}, \bar{s}+1, \ldots, \bar{s}+\bar{p}-1\}$
            $x_m = A_{k-1} u_m$
        end do
        do $m = \bar{s}, \bar{s}+1, \ldots, \bar{s}+\bar{p}-1$
            $y_m = x_m - \sum_{l=\bar{s}}^{m-1} \left[u_l(v_l^T u_m) + v_l(u_l^T u_m)\right]$
            $v_m = y_m - \frac{1}{2} u_m (y_m^T u_m)$
        end do
    end do
    $A_k = A_{k-1} - \sum_{l=s}^{s+p-1} (u_l v_l^T + v_l u_l^T)$
end do
$B = A_N$

first reduction sweep leads to a sufficient performance improvement, because the first reduction sweep is the only part of the algorithm with an algorithmic complexity of $O(n^3)$.

3.3 Application in Material Sciences

The material sciences package WIEN 97 (see Blaha et al. [11], Schwarz, Blaha [25]), used by more than 200 research groups worldwide, allows to compute the electronic properties of solids by using quantum mechanics. One of its computationally most intensive parts is the calculation of about 10 % of the smallest eigenvalues and the corresponding eigenvectors of the real (or complex) Hamiltonian.

Preliminary tests carried out with a fine-grained parallel version of WIEN 97 showed that the symmetric eigensolver was the performance critical part. Due to intricate data dependencies in the tridiagonalization step (see Haunschmid, Kvasnicka [21]) it did not scale well with the number of processors.

Using the independently blocked eigensolvers discussed in this report, significant performance improvements of the parallel version of WIEN 97 can be achieved.

Chapter 4

Experiments

Figures 4.1 to 4.5 and Tables 4.1 and 4.2 illustrate the performance of various parallel eigensolvers, when 10 % of the eigenvalues and eigenvectors are computed. The one-sweep blocked eigensolver of LAPACK, the two-sweep blocked eigensolver, based on the SBR band reduction, and the independently blocked eigensolver, which combines these two blocking methods, have been parallelized by calling the parallel BLAS routines of the optimized SGI/Cray Scientific Library (SCSL). These codes are compared with the SCALAPACK implementation of the one-sweep blocked eigensolver, i. e., a parallelization of the respective LAPACK eigensolver. This SCALAPACK code shows a satisfactory performance behavior for large matrices, where the explicit exchange of messages is an advantage. For small matrices the lower overhead of parallel (SCSL) BLAS leads to better overall performance.

Comparison of the one-sweep, the two-sweep, and the independently blocked eigensolvers, all parallelized by calling parallel BLAS (SCSL in the case of the SGI Cray Origin 2000 or COMPLIB.SGIMATH in the case of the SGI Power Challenge and SGI Power Challenge XL), shows that the independently blocked eigensolver yields the best performance.

4.1 SGI Cray Origin 2000

The SGI Cray Origin 2000 used in the numerical experiments is equipped with 64 MIPS R10000 processors accessing a distributed shared memory of 22 GB.

The MIPS R10000 used in the SGI Cray Origin 2000 is a 64-bit superscalar RISC processor with the following key features:

- 250 MHz CPU clock speed,
- two floating-point operations maximum per cycle, resulting in a theoretical peak performance of 500 Mflop/s, and
- 32 KB two-way set-associative on-chip L1 data cache,
- 32 KB two-way set-associative on-chip L1 instruction cache, and
- 4 MB external L2 cache.

Figure 4.1 and Tables 4.1 and 4.2 reveal the superior parallel performance of the independently blocked algorithm when compared with both singly blocked versions of the two-sweep and the one-sweep blocked algorithm.

A significant observation is the performance decrease of the one-sweep blocked algorithm for matrix sizes between 1500 and 2000. This is due to cache misses in the unavoidable matrix-vector products. The two-sweep blocked algorithm and the independently blocked algorithm show strictly monotonous increasing performance. For large matrices the ScaLapack implementation of the one-sweep blocked algorithm is superior to the one-sweep blocked implementation using parallel Blas. Implementations of the two-sweep blocked algorithm and the independently blocked algorithm in ScaLapack style are not yet available. They should yield even higher performance.

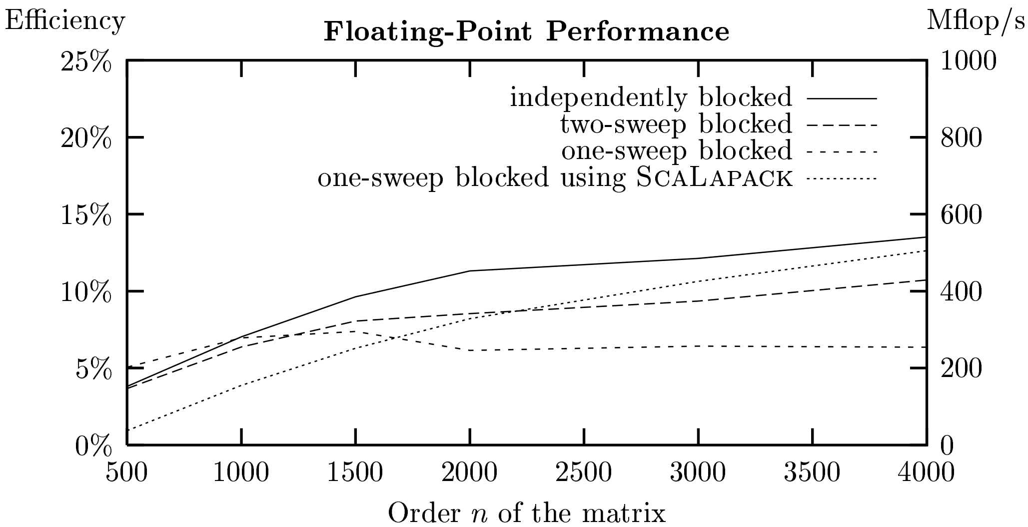

Figure 4.1: Performance of symmetric eigensolvers on 8 MIPS R10000 processors of an SGI Cray Origin 2000. 10 % of the eigenvalues and eigenvectors are computed. Efficiency is given in terms of the theoretical peak performance of 4 Gflop/s.

Table 4.1: Run time (in seconds) of symmetric eigensolvers on one processor of an SGI Cray Origin 2000. 10 % of the eigenvalues and eigenvectors are computed.

$\mathbf{P = 1}$	n	independently blocked	two-sweep blocked	one-sweep blocked	ScaLapack
	1000	7.1 s	7.8 s	7.0 s	10.5 s
	1500	21.3 s	27.2 s	23.9 s	35.3 s
	2000	60.4 s	65.8 s	67.8 s	83.6 s
	3000	210.5 s	245.6 s	249.6 s	296.3 s
	4000	721.9 s	896.3 s	586.4 s	734.0 s

Table 4.2: Run time (in seconds) of symmetric eigensolvers on eight processors of an SGI Cray Origin 2000. 10 % of the eigenvalues and eigenvectors are computed.

$\mathbf{P = 8}$	n	independently blocked	two-sweep blocked	one-sweep blocked	ScaLapack
	1000	5.7 s	6.3 s	5.7 s	10.3 s
	1500	14.0 s	16.8 s	18.3 s	21.5 s
	2000	28.3 s	37.5 s	52.1 s	39.0 s
	3000	89.0 s	115.4 s	167.8 s	101.6 s
	4000	189.4 s	238.6 s	402.4 s	202.6 s

4.2 SGI Power Challenge XL

The SGI Power Challenge XL R10000 used in the numerical experiments is equipped with 16 MIPS R10000 microprocessors that share a common main memory of 4GB.

The MIPS R10000 microprocessor used in the SGI Power Challenge XL is a 64-bit superscalar RISC processor with the following key features:

- 195 MHz CPU clock speed,
- two floating-point operations maximum per cycle, resulting in a theoretical peak performance of 390 Mflop/s,
- 32 KB two-way set-associative on-chip L1 data cache,
- 32 KB two-way set-associative on-chip L1 instruction cache, and
- 2 MB external L2 cache.

The SGI Power Challenge XL utilizes the same processor (with lower clock rate) as the SGI Cray Origin 2000. However, the memory throughput is lower. As a result, Figure 4.2 reveals an even better performance of Level 3 BLAS based algorithms compared with the one-sweep blocked algorithm.

Moreover, the speedup of the one-sweep blocked algorithm is negligible (see Figure 4.4), whereas the speedup of the other two algorithms is much better. Figure 4.3 shows that the speedup of the independently blocked algorithm and of the two-sweep algorithm improves with increasing matrix size. So for larger matrices even more processors could be utilized efficiently.

All speedup numbers are related to the performance of the corresponding algorithms on one processor, not to the best available serial algorithm (which is the independently blocked algorithm on the SGI Power Challenge XL).

Due to memory limitations several curves in Figures 4.2 and 4.3 do not extend to $n_{\max} = 4000$. The current prototype implementation of the two-sweep blocked algorithm requires more memory than the other two algorithms.

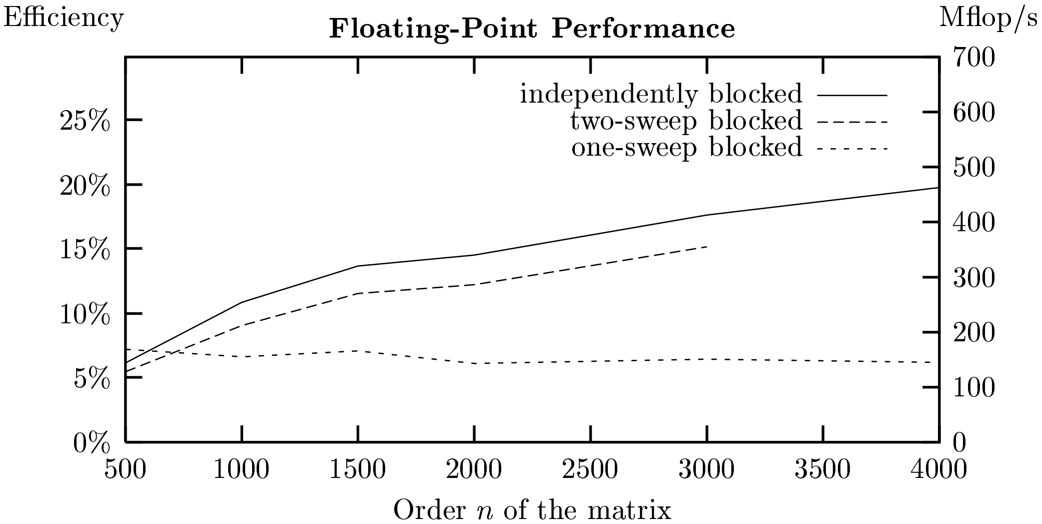

Figure 4.2: Performance of symmetric eigensolvers on 6 MIPS R10000 processors of an SGI Power Challenge XL. 10 % of the eigenvalues and eigenvectors are computed. Efficiency is given in terms of the theoretical peak performance of 2340 Mflop/s.

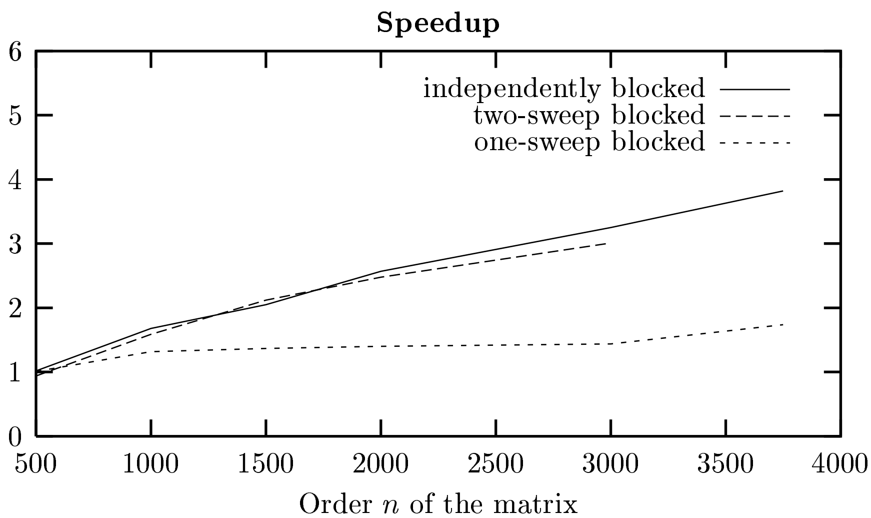

Figure 4.3: Speedup of symmetric eigensolvers on 6 MIPS R10000 processors of an SGI Power Challenge XL with varying matrix size. 10 % of the eigenvalues and eigenvectors are computed.

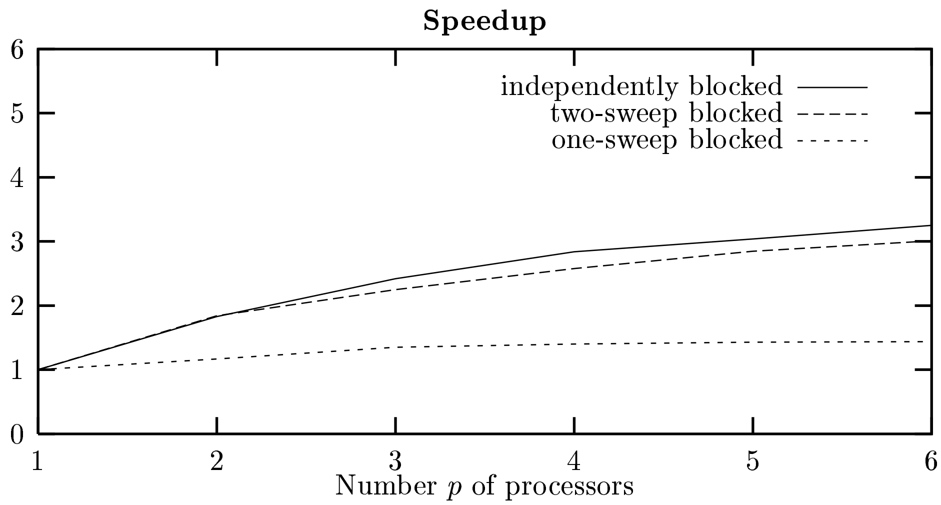

Figure 4.4: Speedup of symmetric eigensolvers on an SGI Power Challenge XL with varying number of processors (MIPS R10000). 10 % of the eigenvalues and eigenvectors are computed. The order of the matrix is $n = 3000$.

4.3 SGI Power Challenge

The SGI Power Challenge used in the numerical experiments is equipped with 6 MIPS R8000 microprocessors that share a common main memory of 768 MB.

The MIPS R8000 microprocessor used in the SGI Power Challenge is a 64-bit superscalar RISC processor with the following key features:

- 90 MHz CPU clock speed,
- four floating-point operations maximum per cycle, resulting in a theoretical peak performance of 360 Mflop/s,
- 32 KB two-way set-associative on-chip L1 data cache,
- 32 KB two-way set-associative on-chip L1 instruction cache, and
- 4 MB external L2 cache.

The MIPS R8000 processor of the SGI Power Challenge has some disadvantages compared with the MIPS R10000. The first level cache of the SGI Power Challenge cannot be utilized by the floating-point units, and the integer performance of this processor is significantly lower because of the lower clock rate.

Matrix-matrix operations are blocked very efficiently in Level 2 cache, nevertheless, but the additional work performed in the two-sweep blocked algorithm and the independently blocked algorithm is a severe obstacle for the overall performance. Still, despite these limitations, the independently blocked algorithm is best also on the SGI Power Challenge (see Figure 4.5).

Due to memory limitations the curves in Figure 4.5 do not extend to $n_{\max} = 4000$.

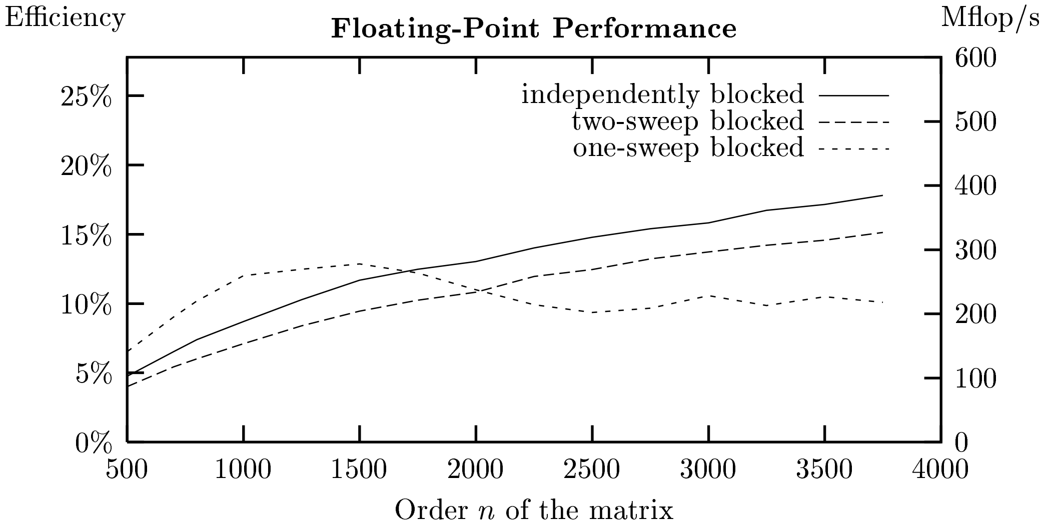

Figure 4.5: Performance of symmetric eigensolvers on 6 MIPS R8000 processors of an SGI Power Challenge. 10 % of the eigenvalues and eigenvectors are computed. Efficiency is given in terms of the theoretical peak performance of 2160 Mflop/s.

Chapter 5

Conclusion

In this report methods for solving symmetric eigenproblems have been discussed, which are efficient in cases where not all eigenvalues and eigenvectors are required. Independently blocked tridiagonalization enables a better exploitation of modern hierarchical memories, and improves scalability on parallel computers.

Bibliography

[1] R. J. Allan, I. J. Bush, *Parallel Application Software on High Performance Computers. Parallel Diagonalisation Routines.*, Technical report, Council for the Central Laboratory of the Research Councils (CCLRC) HPCI Centre at Daresbury Laboratory, available from `http://www.dl.ac.uk/TCSC/HPCI/`, 1996.

[2] E. Anderson et al., LAPACK *Users' Guide, 2nd ed.*, SIAM Press, Philadelphia, 1995.

[3] L. Auslander, A. Tsao, *On parallelizable eigensolvers*, Adv. Appl. Math. 13 (1992), pp. 253–261.

[4] C. H. Bischof, S. Huss-Ledermann, X. Sun, A. Tsao, T. Turnbull, *Parallel Studies of the Invariant Subspace Decomposition Approach for Banded Symmetric Matrices*, Proceedings of Seventh SIAM Conference of Parallel Processing for Scientific Computing, SIAM Press, Philadelphia, 1995, pp. 516–521.

[5] C. H. Bischof, B. Lang, X. Sun, *Parallel Tridiagonalization through Two-Step Band Reduction*, Proceedings of the Scalable High-Performance Computing Conference, IEEE Press, Washington, D. C., 1994, pp. 23–27.

[6] C. H. Bischof, B. Lang, X. Sun, *A Framework for Symmetric Band Reduction*, Technical Report ANL/MCS-P586-0496, Argonne National Laboratory, 1996.

[7] C. H. Bischof, B. Lang, X. Sun, *The* SBR *Toolbox—Software for Successive Band Reduction*, Technical Report ANL/MCS-P587-0496, Argonne National Laboratory, 1996.

[8] C. H. Bischof, M. Marques, X. Sun, *Parallel Bandreduction and Tridiagonalization*, Proceedings of the Sixth SIAM Conference on Parallel

Processing for Scientific Computing, (R. Sincovec, Ed.), SIAM Press, Philadelphia, 1993, pp. 383–390.

[9] L. S. Blackford et al., ScaLapack *Users' Guide*, SIAM Press, Philadelphia, 1997.

[10] P. Blaha, K. Schwarz, P. Sorantin, S. B. Trickey, *Full-Potential, Linearized Augmented Plane Wave Programs for Crystalline Systems*, Comp. Phys. Commun. 59 (1990), pp. 399–415.

[11] P. Blaha, K. Schwarz, J. Luitz, Wien 97. *Improved and Updated* Unix *Version of the Original Copyrighted* Wien *code* [10], Technical report, Vienna University of Technology, 1997.

[12] J. K. Cullum, R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 1: Theory, Volume 2: Programs*, Birkhäuser, Basel, 1985.

[13] I. S. Dhillon, *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue / Eigenvector Problem*, Ph. D. Thesis, University of California, Berkeley, 1997.

[14] J. J. Dongarra, J. Du Croz, S. Hammarling, I. Duff, *A Set of Level 3* Blas, ACM Trans. Math. Software 16 (1990), pp. 1–17.

[15] J. J. Dongarra, J. Du Croz, S. Hammarling, R. J. Hanson, *An Extended Set of* Blas, ACM Trans. Math. Software 14 (1988), pp. 18–32.

[16] J. J. Dongarra, S. J. Hammarling, D. C. Sorensen, *Block reduction of matrices to condensed forms for eigenvalue computations*, J. Comput. Appl. Math. 27 (1989), pp. 215–227.

[17] W. N. Gansterer, D. F. Kvasnicka, C. W. Ueberhuber, *High Performance Computing in Material Sciences. Higher Level* Blas *in Symmetric Eigensolvers*, Technical Report AURORA TR1998-18, Vienna University of Technology, 1998.

[18] W. N. Gansterer, D. F. Kvasnicka, C. W. Ueberhuber, *High Performance Computing in Material Sciences. Numerical Experiments with Symmetric Eigensolvers*, Technical Report AURORA TR1998-19, Vienna University of Technology, 1998.

[19] W. N. Gansterer, D. F. Kvasnicka, C. W. Ueberhuber, *A Level 3 Algorithm for the Symmetric Eigenproblem*, Proceedings of the Third International Meeting on Vector and Parallel Processing (VECPAR '98), Springer-Verlag, Berlin Heidelberg New York Tokio, to appear.

[20] G. Golub, C. Van Loan, *Matrix Computations, 3rd ed.*, The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[21] E. Haunschmid, D. Kvasnicka, *High Performance Computing in Material Sciences. Maximizing Cache Utilization without Increasing Memory Requirements*, Technical Report AURORA TR1998-17, Vienna University of Technology, 1998.

[22] M. Hegland, *A Distribution Independent Algorithm for the Reduction to Tridiagonal Form Using One-Sided Rotations*, First IEEE International Conference on Algorithms and Architectures for Parallel Processing, IEEE Press, Washington, D. C., 1995, pp. 286–289.

[23] B. Hendrickson, E. Jessup, C. Smith, *Towards an Efficient Parallel Eigensolver for Dense Symmetric Matrices*, Technical report, available from `http://www.cs.sandia.gov/~bahendr/papers.html`.

[24] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM Press, Philadelphia, 1998.

[25] K. Schwarz, P. Blaha, *Description of an* LAPW DF *program* (WIEN 95), Quantum-Mechanical Ab-Initio Calculation of Properties of Crystalline Materials, (C. Pisani, Ed.), Vol. 67 of *Lecture Notes in Chemistry*, Springer-Verlag, Berlin Heidelberg New York Tokyo, 1996, pp. 139–154.

[26] R. A. van de Geijn, *Using* PLAPACK*: Parallel Linear Algebra Package*, MIT Press, Cambridge, Massachusetts, 1997.

[27] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

[28] Y.-J. J. Wu, P. A. Alpatov, C. H. Bischof, R. A. van de Geijn, *A Parallel Implementation of Symmetric Band Reduction Using* PLAPACK, Proceedings of the Scalable Parallel Libraries Conference, available from `http://www-Unix.mcs.anl.gov/~jwu/umcp/paper.html`, 1996.