Multi-Sweep Algorithms for the Symmetric Eigenproblem

Wilfried N. Gansterer[1], Dieter F. Kvasnicka[2], and Christoph W. Ueberhuber[3]

[1] Institute for Applied and Numerical Mathematics,
University of Technology, Vienna
ganst@aurora.tuwien.ac.at

[2] Institute for Technical Electrochemistry,
University of Technology, Vienna
dieter@titania.tuwien.ac.at

[3] Institute for Applied and Numerical Mathematics,
University of Technology, Vienna
christof@uranus.tuwien.ac.at

Abstract. This paper shows how the symmetric eigenproblem, which is the computationally most demanding part of numerous scientific and industrial applications, can be solved much more efficiently than by using algorithms currently implemented in Lapack routines.
The main techniques used in the algorithm presented in this paper are (i) sophisticated blocking in the tridiagonalization, which leads to a two-sweep algorithm; and (ii) the computation of the eigenvectors of a band matrix instead of a tridiagonal matrix.
This new algorithm improves the locality of data references and leads to a significant improvement of the floating-point performance of symmetric eigensolvers on modern computer systems. Speedup factors of up to four (depending on the computer architecture and the matrix size) have been observed.

Keywords: Numerical Linear Algebra, Symmetric Eigenproblem, Tridiagonalization, Performance Oriented Numerical Algorithm, Blocked Algorithm

1 Introduction

Reducing a dense symmetric matrix A to tridiagonal form T is an important preprocessing step in the solution of the symmetric eigenproblem. Lapack (Anderson et al. [1]) provides a blocked tridiagonalization routine whose memory reference patterns are not optimal on modern computer architectures. In this Lapack routine a significant part of the computation is performed by calls to Level 2 Blas. Unfortunately, the ratio of floating-point operations to data movement of Level 2 Blas is not high enough to enable efficient reuse of data that

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

reside in cache or local memory (see Table 1). Thus, software construction based on calls to Level 2 BLAS routines is not well suited to computers with a memory hierarchy and multiprocessor machines.

Table 1. Ratio of floating-point operations to data movement for three closely related operations from the Level 1, 2, and 3 BLAS (Dongarra et al. [6])

BLAS	Routine	Memory Accesses	Flops	Flops per Memory Access
Level 1	`daxpy`	$3n$	$2n$	$2/3$
Level 2	`dgemv`	n^2	$2n^2$	2
Level 3	`dgemm`	$4n^2$	$2n^3$	$n/2$

Bischof et al. [4, 5] recently developed a general framework for reducing the bandwidth of symmetric matrices in a multi-sweep manner, which improves data locality and allows for the use of Level 3 BLAS instead of Level 2 BLAS. Their multi-sweep routines improve efficiency of symmetric eigensolvers if only eigenvalues have to be computed. Accumulating the transformation information for the eigenvectors incurs an overhead which can outweigh the benefits of a multi-sweep reduction.

In this paper we introduce an important modification of this framework in case eigenvectors have to be computed, too: We compute the required eigenvectors directly from the intermediate band matrix. Analyses and experimental results show that our approach to improving memory access patterns is superior to established algorithms in many cases.

2 Eigensolver with Improved Memory Access Patterns

A real symmetric $n \times n$ matrix A can be factorized as

$$A = VBV^T = QTQ^T = Z\Lambda Z^T \tag{1}$$

where V, Q, and Z are orthogonal matrices. B is a symmetric band matrix with band width $2b + 1$, T is a symmetric tridiagonal matrix ($b = 1$), and Λ is a diagonal matrix whose diagonal elements are the eigenvalues of the (similar) matrices A, B and T. The column vectors of Z are the eigenvectors of A.

LAPACK reduces a given matrix A to tridiagonal form T by applying Householder similarity transformations. A bisection algorithm is used to compute selected[1] eigenvalues of T, which are also eigenvalues of A. The eigenvectors of T are found by inverse iteration on T. These eigenvectors have to be transformed into the eigenvectors of A using the transformation matrix Q.

[1] If *all* eigenvalues are required, LAPACK provides other algorithms.

Following the multi-sweep framework of Bischof et. al [4, 5], the new method proposed does not compute the tridiagonal matrix T from A directly, but derives a band matrix B as an intermediate result. This band reduction can be organized with good data locality, which is critical for high performance on modern computer architectures. Using a block size b in the first reduction sweep results in a banded matrix B with a semibandwidth of at least b. This relationship leads to a tradeoff:

- If smaller values of b are chosen, then more elements of A are eliminated in the first reduction sweep. This decrease in the number of non-zero elements leads to a smaller amount of data to be processed in later reduction sweeps.
- If larger values of b are chosen, then the Level 3 Blas routines operate on larger matrices and thus better performance improvements are obtained in the first reduction sweep.

Using appropriate values of b speedups of up to ten can be achieved with the two-sweep reduction as compared with the Lapack tridiagonal reduction (see Gansterer et al. [7, 8]).

Eigenvectors can be computed by inverse iteration either from A, B, or T. In numerical experiments inverse iteration on B turned out to be the most effective. The eigenvectors of A have to be computed from the eigenvectors of B using the orthogonal transformation matrix V.

The new algorithm includes two special cases:

1. If $V = Q$ then $B = T$, and the inverse iteration is performed on the tridiagonal matrix T. This variant coincides with the Lapack algorithm.
2. If $V = I$ then $B = A$, and the inverse iteration is performed on the original matrix A. This variant is to be preferred if only a few eigenvectors have to be computed and the corresponding eigenvalues are known.

3 The New Level 3 Eigensolver

The blocked Lapack approach puts the main emphasis on accumulating several elimination steps. b rank-2 updates (each of them a Level 2 Blas operation) are aggregated to form one rank-$2b$ update and hence one Level 3 Blas operation. However, this approach does not take into account memory access patterns in the updating process. We tried to introduce blocking in a much stricter sense, namely by localizing data access patterns.

Partitioning matrix A into blocks consisting of b columns, and further partitioning each column block into $b \times b$ submatrices makes it possible to process the block-columns and their submatrices the same way that the single columns and their elements are processed in the original unblocked method: all *blocks*[2] below the subdiagonal *block* are eliminated, which leaves a *block* tridiagonal matrix, i. e., a *band* matrix (when considered elementwise). At first sight this matrix has bandwidth $4b - 1$. Further examination shows that the first elimination sweep

[2] Replace *block* by *element* (or *elementwise*) to get the original, unblocked, algorithm.

can be organized such that the subdiagonal blocks in the block tridiagonal matrix are of upper triangular form. Hence the band width of B can be reduced to $2b+1$.

Two features distinguish the newly developed Level 3 algorithm from standard algorithms, as, for example, implemented in LAPACK (see Fig. 1):

Tridiagonalization in two sweeps: The first sweep reduces the matrix A to a band matrix B. The second sweep reduces B to a tridiagonal matrix T. No fill-in occurs in the first reduction sweep. In the second sweep, however, additional operations are needed to remove fill-in (see Bischof et al. [4]).

Inverse iteration on the band matrix: Calculating the eigenvectors of B avoids the large overhead entailed by the backtransformation of the eigenvectors of T. It is crucial that the overhead caused by inverse iteration on B instead of T does not outweigh the benefits of this approach.

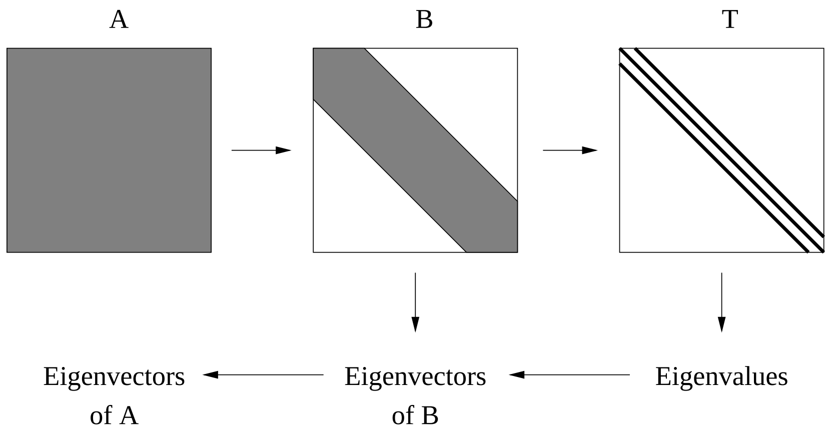

Fig. 1. Basic concept of the new eigensolver

4 Implementation Details

The resulting algorithm has five steps.

1. Reduce the matrix A to the band matrix B.
   (a) Compute the transformation

$$V_i A V_i^T = (I - u_b u_b^T) \cdots (I - u_1 u_1^T) A (I - u_1 u_1^T) \cdots (I - u_b u_b^T)$$

   for a properly chosen subblock of A. Because each Householder vector u_i eliminates only elements below subdiagonal b in the current column

i, the corresponding update from the right does not affect the current column block. Therefore, the Householder vectors for the transformation of the columns of the current column block do not depend on each other. This independence enables a new way of blocking in Step 1c.

   (b) Collect the Householder vectors as column vectors in an $n \times b$ matrix Y and compute the $n \times b$ matrix W such that the transformation matrix V_i is represented as $I - WY^T$ (see Bischof, Van Loan [3]). Matrix Y can be stored in those parts of A which have just been eliminated. Matrix W requires separate storage of order $O(n^2)$; therefore it is overwritten and has to be computed anew in the backtransformation step.
   (c) Perform a rank-$2b$ update using the update matrix $(I - WY^T)$.
   (d) Iterate Steps 1a to 1c over the entire matrix A.
2. Reduce the matrix B to the tridiagonal matrix T. A significant portion of the flops in this step is required to remove fill-in.
3. Compute the desired eigenvalues of the tridiagonal matrix T.
4. Compute the corresponding eigenvectors of B by inverse iteration. The computation of the eigenvectors of B requires more operations than the computation of the eigenvectors of T.
5. Transform the eigenvectors of B to the eigenvectors of the input matrix A. The update matrices W from Step 1(b) have to be computed anew. The transformation matrix V, which occurs in Equation (1), is never computed explicitly.

Variations of the representation of V_i are $V_i = I - YUY^T$ (see Schreiber, Van Loan [10]) and $V_i = I - GG^T$ (see Schreiber, Parlett [9]).

The new algorithm and its variants make it possible to use Level 3 BLAS in all computations involving the original matrix, in contrast to the LAPACK routine `LAPACK/dsyevx`, which performs 50 % of the operations needed for the tridiagonalization in Level 2 BLAS.

5 Complexity

The total execution time T of our algorithm consists of five parts:

$T_1 \sim c_1(b)n^3$ for reducing the symmetric matrix A to a band matrix B,
$T_2 \sim c_2(b)n^2$ for reducing the band matrix B to a tridiagonal matrix T,
$T_3 \sim c_3 kn$ for computing k eigenvalues of T,
$T_4 \sim c_4 kb^2 n$ for computing the corresponding k eigenvectors[3] of B ($n \gg b$), and
$T_5 \sim c_5 kn^2$ for transforming the eigenvectors of B into the eigenvectors of A.

The parameters c_1 and c_2 depend on the semibandwidth b. The parameter c_1 *decreases* in b, whereas c_2 *increases* in b due to an increasing number of operations to be performed. c_3, c_4, and c_5 are independent of the problem size.

[3] If eigenvalues are clustered and reorthogonalization is required, an additional term $\hat{c}_4 k^2 n$ dominates.

The first reduction step is the only part of the algorithm requiring an $O(n^3)$ effort. Thus, a large semibandwidth b of B seems to be desirable. However, b should be chosen appropriately not only to speed up the first band reduction (T_1), but also to make the tridiagonalization (T_2) and the eigenvector computation (T_4) as efficient as possible.

For example, on an SGI Power Challenge the calculation of $k = 200$ eigenvalues and eigenvectors of a symmetric 2000×2000 matrix requires a total execution time $T = 90$ s. T_1 is 57 s, i. e., 63 % of the total time. With increasing matrix size this percentage increases.

6 Results

A first implementation of our algorithm uses routines from the symmetric band reduction toolbox (SBR; see Bischof et al. [2, 4, 5]), EISPACK routines (Smith et al. [11]), LAPACK routines (Anderson et al. [1]), and some of our own routines.

In numerical experiments we compare the well established LAPACK routine `LAPACK/dsyevx` with the new algorithm. On an SGI Power Challenge (with a MIPS R8000 processor running with 90 MHz), speedup factors of up to 4 were observed (see Table 2 and Fig. 5).

Table 2. Execution times (in seconds) on an SGI Power Challenge. $k = n/10$ of the n eigenvalues and eigenvectors were computed

n	k	b	LAPACK `dsyevx`	New Method	Speedup
500	50	6	1.5 s	2.1 s	0.7
1000	100	6	16.6 s	12.6 s	1.3
1500	150	6	74.4 s	39.2 s	1.9
2000	200	10	239.2 s	89.7 s	2.7
3000	300	12	945.5 s	286.4 s	3.3
4000	400	12	2432.4 s	660.5 s	3.7

Fig. 2 shows the normalized computing time $T(n)/n^3$. The significant speedup of the new algorithm when applied to large problems is striking. This speedup has nothing to do with complexity, which is nearly identical for both algorithms (see Fig. 3). The reason for the good performance of the new algorithm is its significantly improved utilization of the computer's potential peak performance (see Fig. 4). The deteriorating efficiency of the LAPACK routine (due to cache effects) causes the $O(n^4)$ behavior of its computation time between $n = 500$ and $n = 2000$ (see Fig. 2).

The new algorithm shows significant speedups compared to existing algorithms for a small subset of eigenvalues and eigenvectors ($k = n/10$) of large matrices ($n \geq 1500$) on all computers at our disposal, including workstations of DEC, HP, IBM, and SGI.

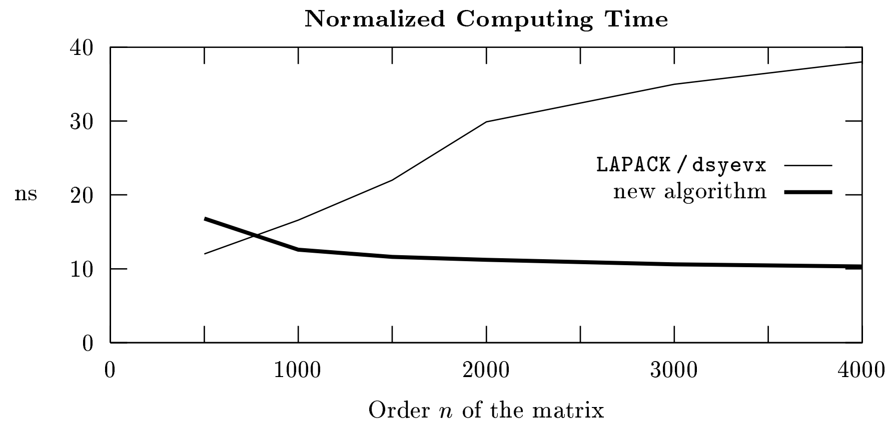

Fig. 2. Normalized computing time $T(n)/n^3$ in nanoseconds

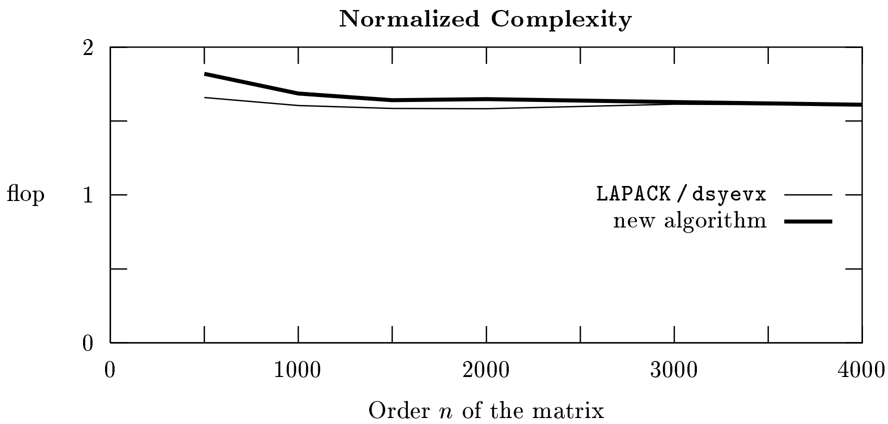

Fig. 3. Normalized number $\mathrm{op}(n)/n^3$ of floating-point operations

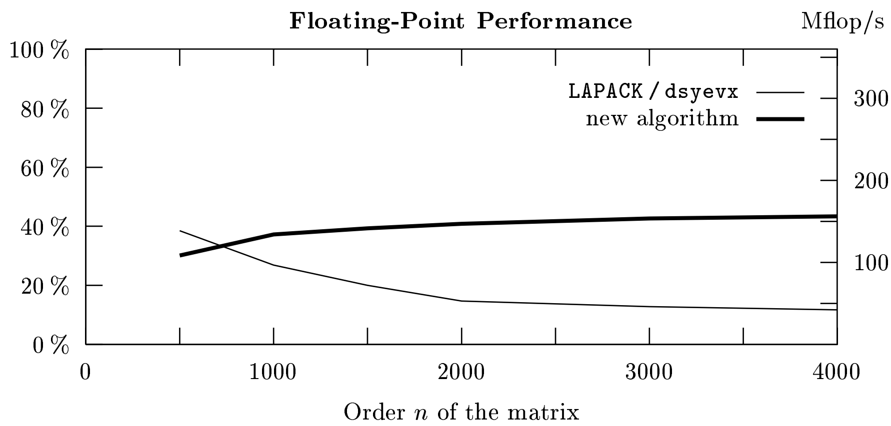

Fig. 4. Floating-point performance (MFlop/s) and efficiency (%)

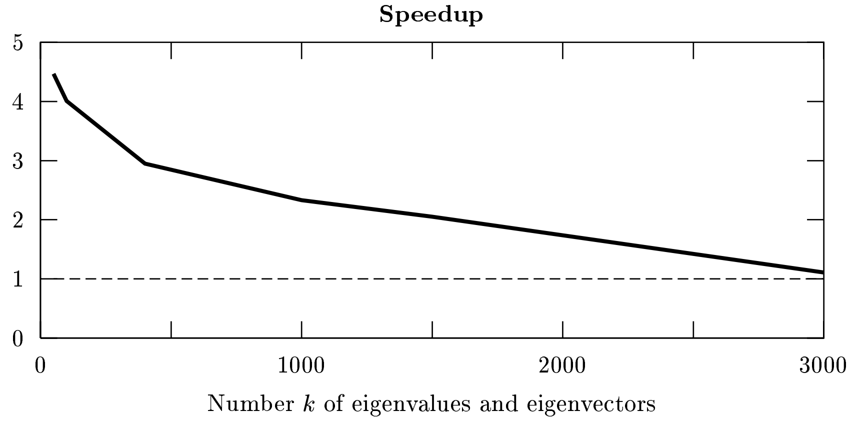

Fig. 5. Speedup of the new algorithm as compared with the LAPACK routine `dsyevx`. Performance improvements are all the better if k is only a small percentage of the $n = 3000$ eigenvalues and eigenvectors

Choosing the block size b. Experiments show that the optimum block size b, which equals the smallest possible band width, increases slightly with larger matrix sizes (see Table 2). A matrix of order 500 requires a b of only 4 or 6 for optimum performance, depending on the architecture (and the size of cache lines). Band widths of 12 or 16 are optimum on most architectures when the matrix order is larger than 2000. A hierarchically blocked version of the new eigensolver which is currently under development allows increasing the block size of the update without increasing the band width and therefore leads to even higher performance.

7 Conclusion

We presented an algorithm for the computation of selected eigenpairs of symmetric matrices which is significantly faster than existing algorithms as, for example, implemented in LAPACK. This speedup is achieved by improved blocking in the tridiagonalization process, which significantly improves data locality.

LAPACK algorithms for the symmetric eigenproblem spend up to 80 % of their execution time in Level 2 BLAS, which do not perform well on cache-based and multiprocessor computers. In our algorithm all performance relevant steps make use of Level 3 BLAS.

The price that has to be paid for the improved performance in the tridiagonalization process is that the eigenvectors cannot be computed from the tridiagonal matrix because of prohibitive overhead in the backtransformation. They have to be computed from an intermediate band matrix.

When choosing the block size, a compromise has to be made: Larger block sizes improve the performance of the band reduction. Smaller block sizes reduce the band width and, therefore, speed up the inverse iteration on the band matrix.

The new algorithm is particularly attractive (on most architectures) if not all eigenvectors are required.

If the gap between processor speed and memory bandwidth further increases, our algorithm will be highly competitive also for solving problems where all eigenvectors are required.

Future Development. Routines dominated by Level 3 BLAS operations, like the eigensolver presented in this paper, have the potential of speeding up almost linearly on parallel machines. That is why we are currently developing a parallel version of the new method. Another promising possibility of development is the use of hierarchical blocking (see Ueberhuber [12]).

References

1. E. Anderson et al., LAPACK *Users' Guide, 2nd ed.*, SIAM Press, Philadelphia, 1995.
2. 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.
3. C. H. Bischof, C. F. Van Loan, *The WY Representation for Products of Householder Matrices*, SIAM J. Sci. Comput. 8 (1987), pp. s2–s13.
4. C. H. Bischof, B. Lang, X. Sun, *A Framework for Symmetric Band Reduction*, Technical Report ANL/MCS-P586-0496, Argonne National Laboratory, 1996.
5. 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.
6. J. J. Dongarra, I. S. Duff, D. C. Sorensen, H. A. van der Vorst, *Linear Algebra and Matrix Theory*, SIAM Press, Philadelphia, 1998.
7. W. Gansterer, D. Kvasnicka, C. Ueberhuber, *High Performance Computing in Material Sciences. Higher Level* BLAS *in Symmetric Eigensolvers*, Technical Report AURORA TR1998-18, Vienna University of Technology, 1998.
8. W. Gansterer, D. Kvasnicka, C. Ueberhuber, *High Performance Computing in Material Sciences. Numerical Experiments with Symmetric Eigensolvers*, Technical Report AURORA TR1998-19, Vienna University of Technology, 1998.
9. R. Schreiber, B. Parlett, *Block Reflectors: Theory and Computation*, SIAM J. Numer. Anal. 25 (1988), pp. 189–205.
10. R. Schreiber, C. Van Loan, *A Storage-Efficient WY Representation for Products of Householder Transformations*, SIAM J. Sci. Stat. Comput. 10–1 (1989), pp. 53–57.
11. B. T. Smith et al., *Matrix Eigensystem Routines*—EISPACK *Guide*, Lecture Notes in Computer Science, Vol. 6, Springer-Verlag, Berlin Heidelberg New York Tokyo, 1976.
12. C. W. Ueberhuber, *Numerical Computation*, Springer-Verlag, Berlin Heidelberg New York Tokyo, 1997.