A Framework for Symmetric Band Reduction and Tridiagonalization [1]

Christian H. Bischof
bischof@mcs.anl.gov
Xiaobai Sun
xiaobai@mcs.anl.gov

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439–4801

Preprint MCS–P298–0392

Abstract

This paper develops a framework for bandwidth reduction and tridiagonalization algorithms for symmetric banded matrices. The algorithm family includes the algorithms by Rutishauser and Schwarz, which underly the EISPACK and LAPACK implementations, and the algorithm recently proposed by Lang. Our framework leads to algorithms that require fewer floating-point operations, allow for space-time tradeoffs, enable the use of block orthogonal transformations, and increase the degree of parallelism inherent in the algorithm.

1 Introduction

Reduction to tridiagonal form is a major step in eigenvalue computations for symmetric matrices. If the matrix is full, the conventional Householder tridiagonalization approach [11, 4, p. 276] or block variants thereof [9] is the method of choice.

However, for banded matrices with semibandwidth b, where $b \ll n$, this approach is not optimal since the matrix being reduced has completely filled in after $log_2(n/(b-1))$ reduction steps. It is well known that the algorithm of Rutishauser [15] and Schwarz [17] (called the R-S algorithm in the rest of the paper) is more economical than the standard approach when b is small compared to n. In the R-S algorithm, elements in the current column to be reduced are annihilated one at a time by Givens rotations. Each Givens rotation generates a fill-in element outside of the original band, and the fill-in is chased out by a sequence of Givens rotations.

In the same paper [15], Rutishauser also suggested a band reduction scheme based on Householder transformations that annihilates all $b-1$ elements in the current column instead of only one, Rutishauser used an analogous chasing scheme to drive out the triangular bulge generated by the reduction with a sequence of Householder transformations, as shown in Figure 1. However, due to

[1] This work was supported by the Applied and Computational Mathematics Program, Defense Advanced Research Projects Agency, under contract DM28E04120, and the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

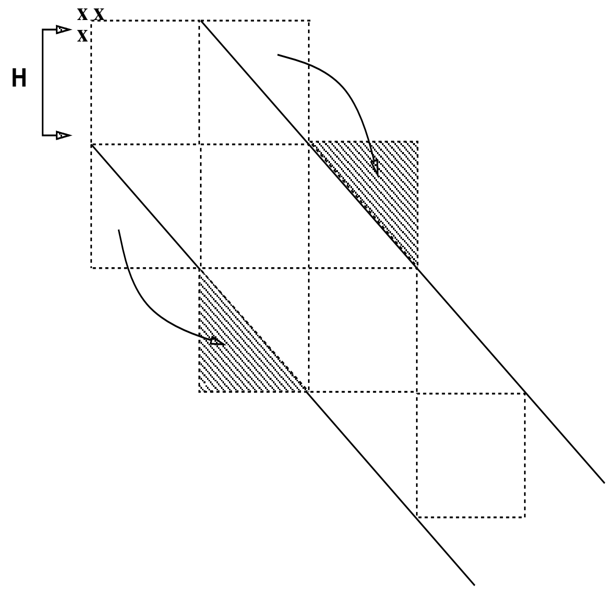

Figure 1: Rutishauser's Tridiagonalization with Householder Transformations

the significant work involved in chasing the triangular bulges, this algorithm does not perform better than the Givens-rotations based R-S algorithm. The R-S algorithm can be vectorized (along the diagonal) [12] and this variant is the basis of the band reduction algorithms in EISPACK [18, 10] and LAPACK [2].

The R-S algorithm requires storage for one extra subdiagonal, and Rutishauser's Householder approach requires storage for $b-1$ extra subdiagonals. To assess the storage requirements of various algorithms, we introduce the concept of *working semibandwidth*. The working semibandwidth of an algorithm for a symmetric band matrix is the number of sub(super)diagonals accessed during its implementation. For instance, the working semibandwidth is $b+1$ for the R-S algorithm and $2b-1$ for Rutishauser's algorithm with Householder transformations.

In both algorithmic approaches, each reduction step has two parts:

- Band reduction (either from b to $b-1$, or b to 1), and
- Bulge Chasing to maintain banded form.

Either way, the bulk of the computation is spent in bulge chasing. Lang's algorithm [13, 14] improves on the bulge chasing strategy. It employs Householder transformations to eliminate all $b-1$ subdiagonal entries in the current column, but instead of chasing out the whole triangular bulge (only to have it reappear the next step) Lang's algorithm chases only the first column of the bulges. These are the columns that (if not removed) would increase working bandwidth in the next step. The working semibandwidth for Lang's algorithm is also $2b-1$. By letting the rest of the bulges stay, Lang's algorithm requires less flops than the R-S algorithm. On the other hand, the R-S algorithm requires less storage and may still be preferable if storage is tight.

In this paper, we generalize the ideas behind the R-S algorithm and Lang's algorithm. We develop a band reduction algorithm which eliminates d subdiagonals of a symmetric banded matrix with semibandwidth b ($b > d$), in a fashion akin to Lang's tridiagonalization algorithm. Then, like the R-S algorithm, the band reduction algorithm is repeatedly used until the reduced matrix is tridiagonal. If $d = b-1$, it is Lang's algorithm; and if d is always chosen to be 1, it results in the R-S algorithm. The freedom we have in choosing d leads to a class of algorithms for banded reduction and tridiagonalization with favorable computational properties. In particular, we can derive:

1. algorithms with minimum algorithmic complexity,
2. algorithms with minimum algorithmic complexity subject to limits on available storage,
3. algorithms with increased scope for parallel computation, and
4. algorithms employing BLAS-3 kernels through block orthogonal reductions.

The paper is organized as follows. In next section, we introduce our framework for band reduction and tridiagonalization of symmetric banded matrices. In Section 3, we derive algorithms that are optimal with respect to either computational cost or space complexity and present some computational results. In Section 4 we give an intuitive understanding why our approach leads to algorithms with a lower computational complexity. We then show that our framework allows for the use of block orthogonal transformations and increases the inherent potential for parallelism in the algorithm. We sum up our findings in Section 6.

2 A Framework for Band Reduction and Tridiagonalization

In this section we describe a framework for band reduction and tridiagonalization of symmetric matrices. We use $H(i_1, i_2, x)$ to denote a Householder matrix

$$H(i_1, i_2, x) = I - \beta v v^{\mathrm{T}}$$

where $i_2 > i_1$ and v is determined such that

$$(Hx)_i = \begin{cases} x_i, & \text{if } i < i_1 \text{ or } i > i_2 \\ \pm\|x(i1 : i2)\|_2 & \text{if } i = i_1 \\ 0 & \text{if } i_1 < i \leq i_2 \end{cases}$$

That is, H annihilates x's components from i_1+1 to i_2. When x is the jth column of a given matrix, the notation $H(i_1, i_2, j)$ is used instead.

Suppose an n-by-n symmetric band matrix with semi-bandwidth b is to be reduced to a band matrix with semi-bandwidth $b - d$, with $1 < d < b$. We assume that at the beginning of each reduction step, the current submatrix to be reduced is of the form shown in Figure 2. In other words:

- The first column and row, i.e., the current column/row, has semibandwidth equal to b,
- The remainder of the submatrix, i.e., the current trailing matrix, is block tridiagonal with all diagonal blocks but the last one being of order b and the last one being of order $\leq b$.
- every subdiagonal block is upper triangular in its first $b - d - 1$ columns.

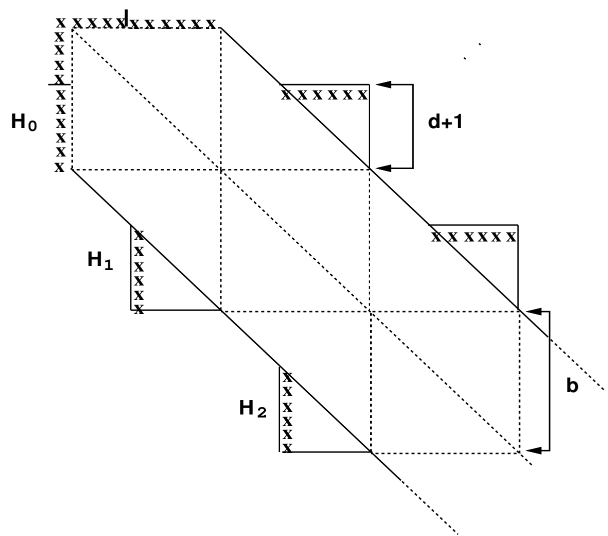

Figure 2: Matrix structure at the entry of each column reduction

This assumption is true for the first step, since the sub(super)-diagonal blocks are upper(lower) triangular initially. Also notice that the current matrix is banded with semi-bandwidth $b - d + 1$ and so is the current working semibandwidth.

Now, let $H_0 = H(b - d + 1, b + 1, 1)$ be the Householder matrix to reduce the semibandwidth in the first column to $b - d$ and $H_i = H(ib + (b - d + 1), ib + (b + 1), (i - 1)b + (b - d + 1))$ be the Householder matrix to chase the bandwidth back to b in the $(b - d)$th column of the ith subdiagonal block. That is, H_i eliminates the first column outside the bth subdiagonal of the "bulge" created by H_i. In the example shown in Figure 2, $b = 10$ and $d = 5$.

We recognize the following points:

1. The transformations do not alter the structure of the current trailing matrix, so the current working bandwidth $b - d$ is kept, between $b + 1$ for the R-S algorithm and $2b - 1$ for Lang's algorithm.
2. Every Householder transformation H_i, $i \geq 0$, involves the same number of rows and columns.
3. The resulting trailing matrix has the structure in Figure 2 for the reduction in next step.

Hence, we arrive at the following algorithm:

Algorithm 2.1 : Band Reduction Algorithm band_rd(n, A, b, d)

input An n-by-n symmetric band matrix A with semibandwidth $b > 1$. An integer d, $1 \leq d < b$, the number of sub(super) diagonals to be eliminated.
output An n-by-n symmetric band matrix with semibandwidth $b - d$.

for $\quad top = 1 : n - (b - d + 1)$

1) band reduction in the first column and row of $A(top : n, top : n)$, which includes generating the Householder matrix H_0 and updating the $d + 1$ rows and columns affected.

2) bulge reduction in the first column or row of every triangular bulge, which includes generating the Householder matrices H_i and updating every $d + 1$ rows and columns affected.

end for

The working semibandwidth of the algorithm is $b + d$. Its arithmetic cost in flops is

$$\begin{aligned} &cost(n, b, d) \approx \\ &\quad \left(4(d + 1) + (2d^2 + 7d + 5.5)/b\right) \\ &\quad \times \left(n(n - 1) - (b - d)(b - d + 1)\right). \end{aligned} \tag{2.1}$$

Given the band reduction algorithm, we can now derive a tridiagonalization algorithm in a straightforward fashion by "peeling off" subdiagonals in chunks.

Algorithm 2.2: Tridiagonalization Algorithm tri_diag(n,A,b,$\{d_i\}$)

input An n-by-n symmetric band matrix with semibandwidth $b > 1$. A sequence of positive integers $\{d_i\}_{i=1}^{k}$ with $\sum d_i = b - 1$.
output An n-by-n symmetric tridiagonal matrix.
initial $b_i = b$
for $\quad i = 1, k$
call band_rd(n, A, b_i, d_i)
$b_i = b_i - d_i$
end for

The working semibandwidth for the tridiagonalization algorithm is $b + d_1$ if the $\{d_i\}$ satisfy

$$d_i \le d_1 + \sum_{j=1}^{j=i-1} d_j, \qquad 1 < i \le k. \tag{2.2}$$

The arithmetic cost is

$$\sum_{i=1}^{k} cost(n, b_i, d_i)$$

where $b_1 = b$ and $b_{i+1} = b_i - d_i$ for $1 \le i < k$.

3 Optimal Tridiagonalization Algorithms

In this section we derive optimal tridiagonalization algorithms. Our problem is to

$$\begin{array}{rl} \textbf{min} & \text{number of flops to tridiagonalize an} \\ & n \times n \text{ banded matrix with bandwidth } b \\ \textbf{subject to:} & \text{working bandwidth} \le sb, \text{where } sb \ge b + 1. \end{array} \tag{3.3}$$

Denote by $\{d_i\}$ a band difference sequence for Algorithm 2.2. Given a limit on sb, we can use dynamic programming [1] to determine an optimal sequence $\{d_i\}$ from the cost function given in (2.1). By taking the storage requirements into account, we allow for space tradeoffs and allow for the most efficient utilization of available memory.

3.1 No Storage Constraints

We first consider Problem (3.3) with $sb \geq 2b - 1$. In this case, we in fact do not have storage constraints, since the maximum working semibandwidth of Algorithm 2.2 is $2b - 1$. For small b, compared to n, Lang's algorithm is optimal. Otherwise, the optimal sequence $\{d_i\}$ is quite different from that of Lang's algorithm.

For example, for a $50,000$-by-$50,000$ symmetric matrix with semibandwidth 300, the optimal sequence is

$$\{26, 25, 24, 23, 22, 20, 19, 18, 17, 16, 15, 14, 12, 11, 10, 9, 8, 10\}, \tag{3.4}$$

the reduction requires 3.48e+12 floating point operations, and the working semibandwidth is 310. In contrast, Lang's algorithm would require 4.47e12 flops, and its working semibandwidth is 599. We also note that using the constant sequence

$$\{16, \ldots, 16, 11\}$$

requires 3.49e+12 flops with a working semibandwidth of 316. A constant sequence

$$\{32, \ldots, 32, 11\}$$

requires 3.56e+12 flops with a working bandwidth of 332. Hence a *constant-stride* sequence seems to be just as good a choice as the optimal one from a practical point of view, and saves the dynamic programming overhead.

3.2 Minimum Storage

Now consider another extreme case of the constraint in Problem (3.3): $sb = b + 1$. That is, we have space for at most one other subdiagonal. Even for small b, Lang's algorithm is not among the candidates since its working semibandwidth is $2b - 1 > b + 1$. Indeed, a candidate d-sequence for this case should satisfy the condition (2.2) with $d_1 = 1$. The sequence of the R-S algorithm is a constant sequence with $d = 1$, which satisfies the conditions. We suggest

$$\begin{aligned} &d_1 = 1, \\ &d_{i+1} = 2d_i, \qquad 1 \leq i < k = \lfloor \log_2(b-1) \rfloor \\ &d_{k+1} = (b-1) - d_k. \end{aligned} \tag{3.5}$$

We call it the *doubling-stride sequence* since the bandwidth reduction size doubles in each round.

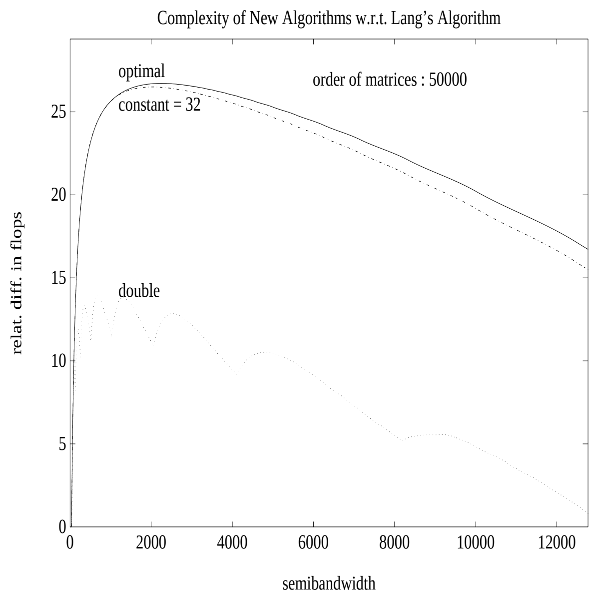

Figure 3: Relative Improvement in Complexity over Lang's Algorithm

3.3 An Example

To get a feeling for the improvement that we could expect from the new algorithms, we computed the complexity of tridiagonalizing a banded matrix of order 50,000. Figure 3 shows the relative decrease in complexity of the doubling-stride and constant-stride algorithms compared to Lang's algorithm for matrices with varying initial bandwidth. We see that the new approaches can save up to a quarter of the floating point operations and that the "doubling-stride" approach has a higher complexity than the "constant" approach.

4 Understanding Optimality

We have seen that the R-S algorithm peels off subdiagonal bands one by one and requires minimum storage, but it is not optimal among algorithms with minimum storage. On the other hand, Lang's eliminates all subdiagonals in one shot, but it is not optimal either when b is large. In this section we try to give an intuitive explanation why the complexity of our scheme is superior to both those approaches.

First, let us compare Lang's algorithm and Algorithm 2.2 with a sequence $\{d, b–d–1\}$. During the bulge chasing following the reduction in column 1, the data area accessed by Lang's algorithm with rank-1 row updating and rank-1 column updating (or shortly, rank-2 updating) is shown in Figure 4. The data area accessed by Algorithm 2.2 with sequence $\{d, b-d-1\}$ with rank-1 row(column) updating or rank-2 updating is shown in Figure 5 and Figure 6. For easier comparison, we now

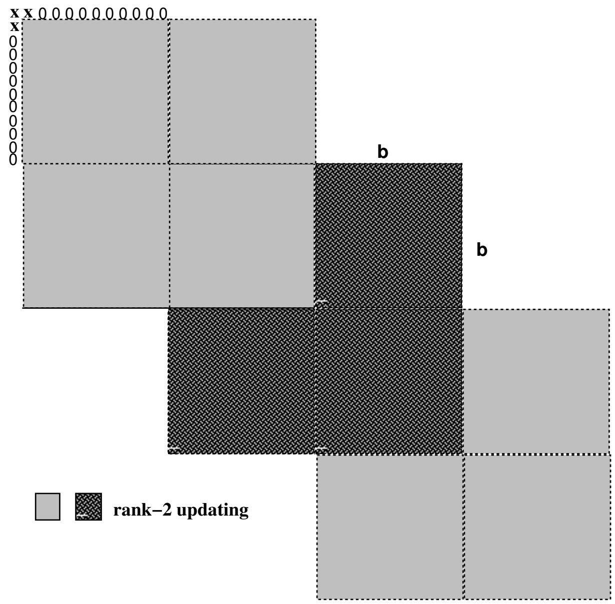

Figure 4: Data area visited by Lang's algorithm with rank-2 updating

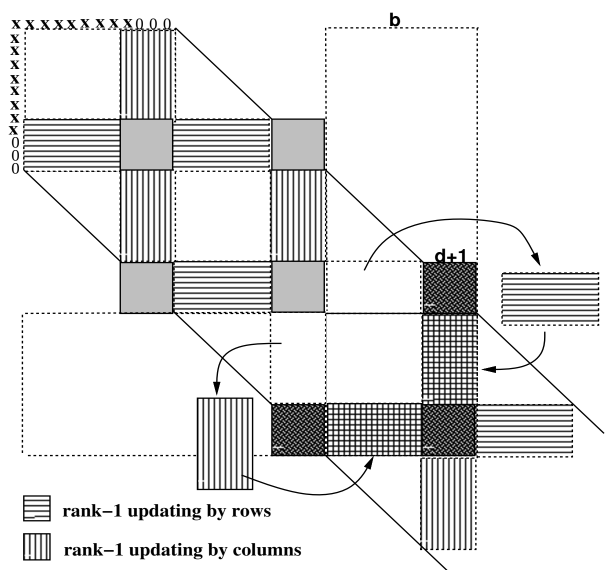

Figure 5: Data area visited by subroutine band_rd(n, A, b, d)

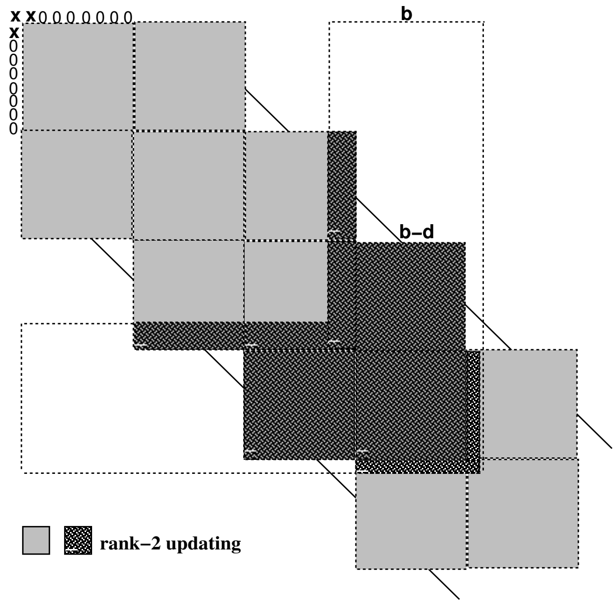

Figure 6: Data area visited by subroutine band_rd$(n, A, b-d, b-d-1)$

take the rank-2 updating as basic computation and examine in a unit area outlined by the dash lines as in Figure 6 the total data area visited by each algorithm for the purpose of rank-2 updating. For Lang's algorithm, it is

$$a_L = 3b^2.$$

For the two successive band reductions, on average it is

$$a_B = 3b^2 + (4d^2 + 3d + 2) - 2b(d-1). \tag{4.1}$$

If d is chosen such that $3 \leq d \leq b/4$, then $a_B < a_L$. That is, the sum of the small areas in Figure 5 and Figure 6 is smaller than that of the big areas in Figure 4. In other words, if b is large enough and d is properly chosen, Algorithm 2.2 with sequence $\{d, b-d-1\}$ requires less floating point operations than Lang's algorithm. Of course, the same idea can then be applied recursively on the tridiagonal reduction for the matrix with reduced semibandwidth $b-d$.

Now let us consider how often a row or column is repeatedly involved in a reduction or chasing step. The data area visited by the R-S algorithm for annihilating 3 elements in the first column in 3 rounds is shown in Figure 7. We see in particular that the total area visited in the first b-by-b block is almost twice as big as the one by Algorithm 2.1 with $d = 3$, due to the revisiting of the first row and column in the last visited area. We shall mention here that the revisiting of the first row and column in the last visited area also interprets the well known fact that the tridiagonal reduction for full matrices by Givens rotation costs twice as much as that by Householder transformation. A way to reduce the revisiting area is to increase the number of rows and columns visited each round by using Householder transformations.

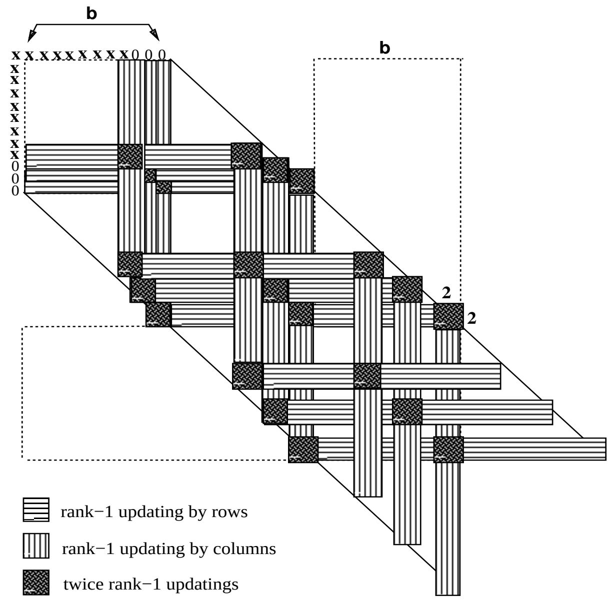

Figure 7: Three rounds of band reduction with the R-S algorithm

So, in comparison to the R-S and Lang's algorithm, our algorithm can be interpreted as balancing the counteracting goals of

- decreasing the number of times a column is revisited (by the use of Householder transformations), and
- decreasing the area involved in updates (by peeling off subdiagonals in several chunks).

5 Scope for Block Orthogonal Transformations and Enhanced Parallelism

Lang's work was in large part motivated by the desire to develop a parallel algorithm for the band tridiagonalization problem. In both Lang's and the R-S algorithm, parallelism can be exploited by pipelining the chasing processes associated with several columns. By using Householder reductions, and an improved bulge chasing strategy (compared to Rutishauser's algorithm) Lang arrived at an algorithm that increases computational granularity to allow for the use of BLAS-2 kernels, while maintaining a degree of parallelism sufficient for a moderate number p of processors [14].

Our approach increases both the inherent parallelism in the algorithm and enables the use of BLAS 3 kernels through the use of block orthogonal transformations. Figure 8 illustrates that we can start the band reduction step with width d_k as soon as the previous band reduction step (with width d_{k-1}) has progressed far enough down the diagonal. This level of parallelism is in addition to the one

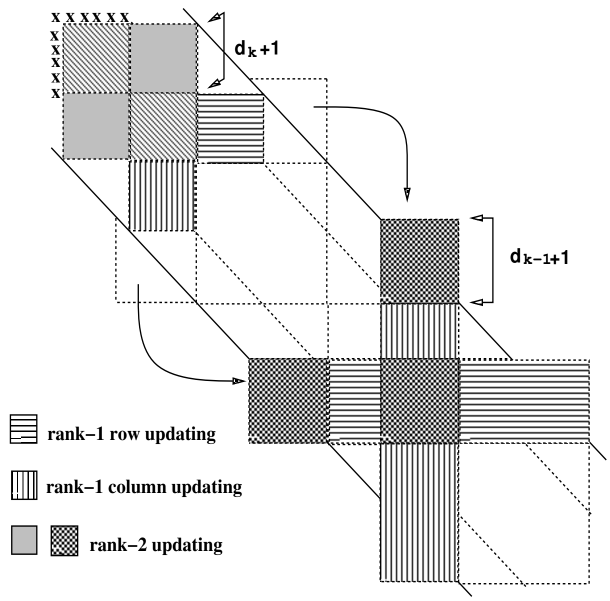

Figure 8: Pipelining Successive Band Reductions

exploited in Lang's algorithm, and the resulting "parallel slackness" allows for an implementation that should be much less sensitive to the ratio between n and b. We mention in particular that this extra degree of parallelism could be favorably exploited even in a tridiagonalization algorithm for dense matrices, where current approaches [6, 8] are limited to exploiting parallelism through pipelining of different reduction steps, or inside the application of a Householder or block Householder reduction.

To develop a block algorithm, we note that, as long as $d < b - 1$, the reduction in the second column can start after H_0 has been applied, since it does not interfere with the bulge chasing associated with the reduction in the first column (see Figure 2). Actually, we can initiate the reduction up to the nb-th column if $nb \leq b-d$, that is, as long as the column to be reduced is ahead of the bulge produced by the first reduction. Doing so then produces the shaded bulge shown in Figure 9, which is then chased with transformations nb Householder transformations. Notice that compared to Algorithm 2.1. only the ordering of the transformations has changed in that they now are applied in groups of nb.

However, in doing so, we have enabled the use of block transformations. For example, we can use the "WY" [5] or "compact WY" [16] representation for the product of Householder matrices to express the Householder updates in closed form and then exploit the speed of matrix-matrix multiply in the application of those transformations. Due to the fact that transformations are applied only to a diagonal subblock (as compared to whole rows or columns in the usual dense schemes), we expect a block algorithm to be superior only for matrices of relatively large bandwidth, and on parallel machines, where block transformations also result in a reduction of message transfer [3, 8].

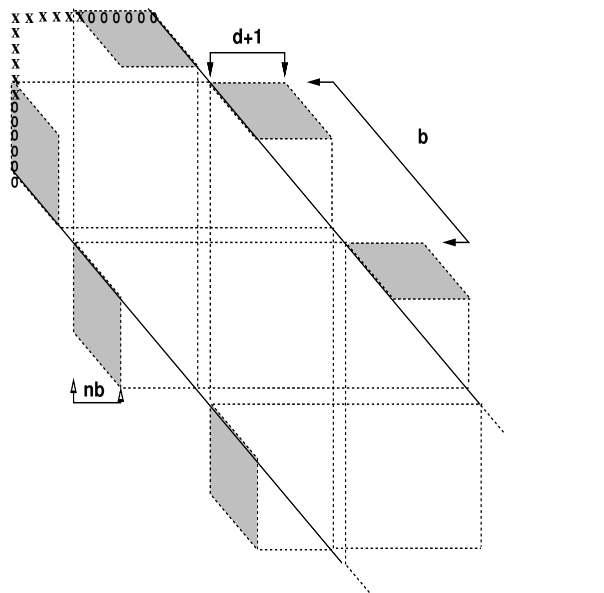

Figure 9: Band reduction and bulge reduction in blocks

6 Experimental Results

While the constant- and doubling-stride algorithms are superior from a complexity point of view, they lead to algorithms with shorter vector lengths, and they require more passes over the data than Lang's algorithm. To assess the effect of this different computing behavior in a workstation environment, we implemented a BLAS-2 version of the constant- and doubling-stride algorithms and compared them with Lang's algorithm. Even though it does not use block orthogonal reductions, the code already does the reduction and bulge chasing in "chunks" of $\min(32, b-d)$ columns in order to increase data locality.

We performed two sets of experiments:

- Matrices of order 1200 and semibandwidth up to 400 on a Sparcstation II
- Matrices of order 2400 and semibandwidth up to 800 on an IBM RS6000/550.

The test matrices were stored in the usual two-dimensional Fortran storage scheme, and in addition to the diagonal only the $2b-1$ subdiagonals (for Lang's algorithm), the lower $b+32$ subdiagonals (for the constant-stride algorithm), and the lower $b+1$ subdiagonals (for the doubling-stride algorithm) of the lower triangle were accessed. On the Sparcstation, we used the Fortran BLAS from netlib [7], on the IBM, we used the vendor-supplied assembler routines. All computations were performed in single precision. Figures 10 and 11 show the relative behavior of the new algorithms with respect to Lang's algorithm as predicted by the complexity analysis and as actually observed. We see that on

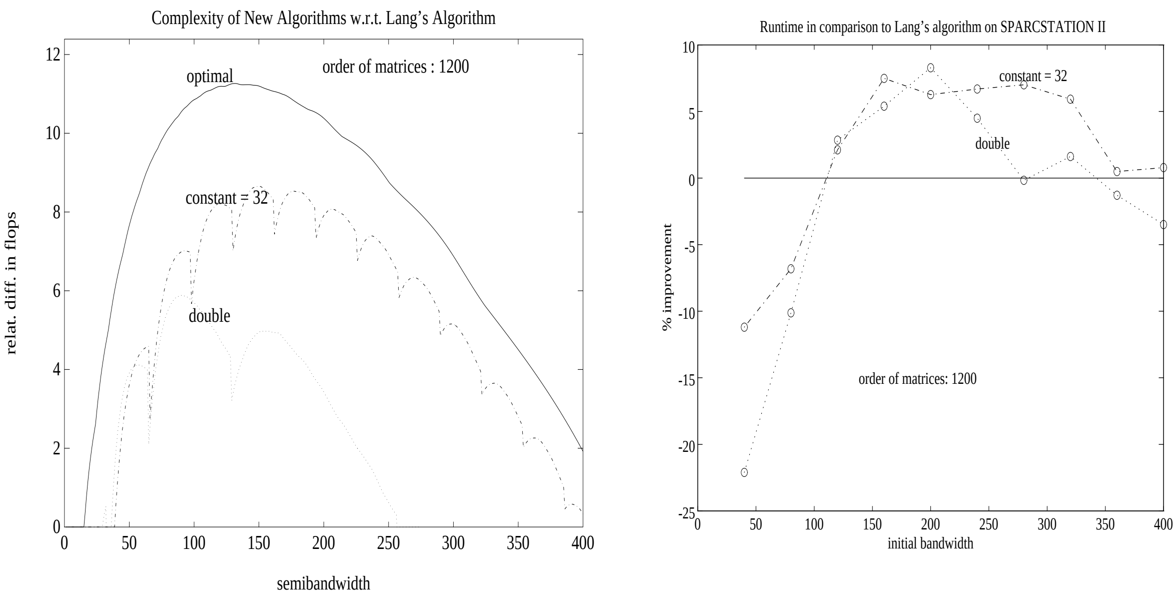

Figure 10: Experimental Results on Sparcstation II

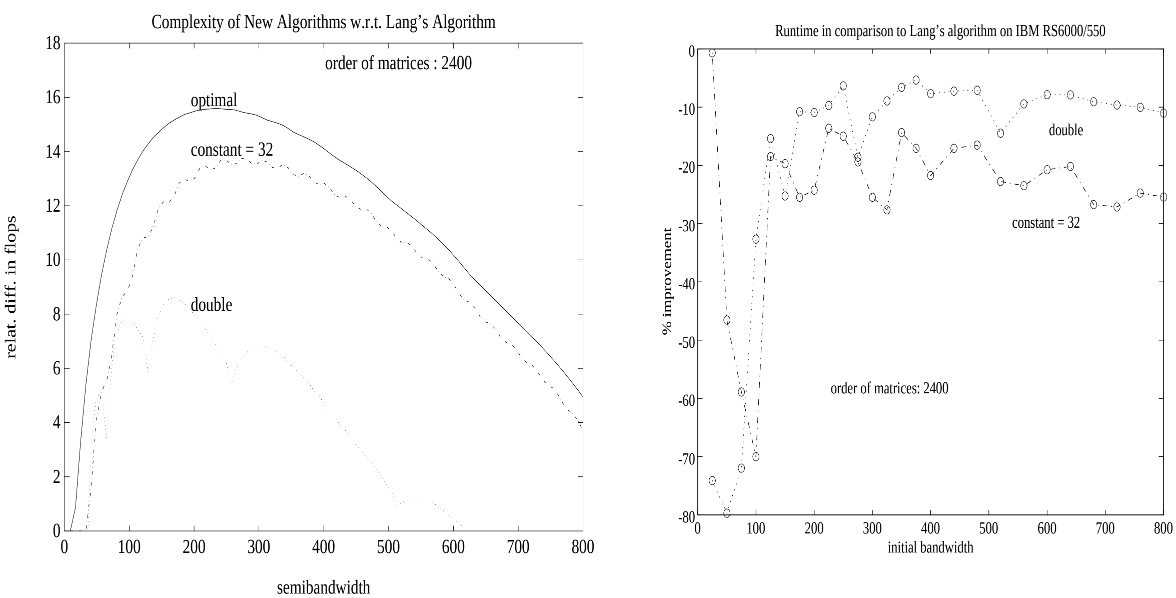

Figure 11: Experimental Results on IBM RS6000/550

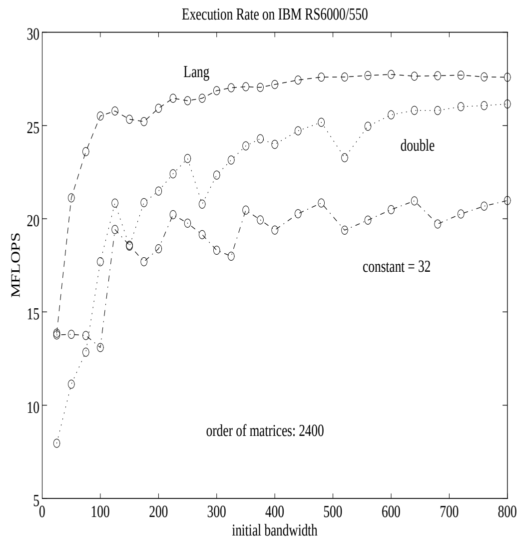

Figure 12: Execution Rate of Tridiagonalization Algorithms on IBM RS6000/550

the Sparcstation II we can in fact outperform Lang's algorithm as the complexity analysis predicted. On the IBM RS6000's cache architecture, however, the increased data movement and the shorter vector length seem to eliminate the flop count advantage, and both new algorithms run slower. This is also reflected in the execution rates, which are shown in Figure 12. In contrast, all algorithms ran around 5.6 Mflops on the Sparcstation.

It is also surprising that on the RS6000 the doubling-stride algorithm performs better than the constant-stride algorithm, despite its complexity disadvantage, and despite the fact that it sweeps over the matrix more often. We also expect that a packed implementation of the new algorithms would perform better in comparison to Lang's on the RS6000, since both algorithms require significantly less memory, and that in itself would improve memory locality.

7 Conclusions

We have introduced a framework for band reduction and tridiagonalization, which generalizes the ideas underlying Rutishauser's, Schwarz' and Lang's algorithm. By "peeling off" subdiagonals in chunks, we arrived at algorithms that require less floating-point operations and less storage. We also provided an intuitive explanation why our approach, which eliminates subdiagonals in groups, has a lower computational complexity than the previous algorithms, which either eliminated subdiagonals one by one or all at once. Our approach allows for the use of block algorithms and increases the degree of parallelism. In particular, it allows for a parallel tridiagonalization algorithm even of a full matrix, that has significantly more "parallel slackness" than previous approaches. Experimental results on workstations do however indicate that at least for the conventional two-dimensional matrix storage scheme Lang's algorithm can still outperform the new approaches. However, for larger matrices and parallel environments we expect a true block version of this algorithm (employing block Householder reductions) to be superior.

Acknowledgements

We would like to thank Jack Dongarra of the University of Tennessee at Knoxville for making the IBM RS6000/550 accessible to us.

References

[1] Alfred Aho, John Hopcroft, and Jeffrey Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass., 1974.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, PA, 1992. to appear.

[3] Christian H. Bischof. *A Pipelined QR Factorization algorithm with Adaptive Blocking*, pages 10–20. Ellis Horwood Publishers, Chichester, U.K., 1989.

[4] Christian H. Bischof and Xiaobai Sun. A divide-and-conquer method for computing complementary invariant subspaces of symmetric matrices. Technical Report MCS–P286–0192, Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[5] Christian H. Bischof and Charles F. Van Loan. The WY representation for products of Householder matrices. *SIAM Journal on Scientific and Statistical Computing*, 8:s2–s13, 1987.

[6] H. Chang, S. Utku, M Salama, and D. Rapp. A parallel Householder tridiagonalization stratagem using scattered square decomposition. *Parallel Computing*, 6:297–311, 1988.

[7] Jack Dongarra and Eric Grosse. Distribution of mathematical software by electronic mail. *Communications of the ACM*, 30(5):403–407, 1987.

[8] Jack Dongarra and Robert van de Geijn. Reduction to condensed form on distributed memory architectures. Technical Report CS-91-130, Computer Science Department, The University of Tennessee, 1991.

[9] Jack J. Dongarra, Sven J. Hammarling, and Danny C. Sorensen. Block reduction of matrices to condensed form for eigenvalue computations. Technical Report MCS–TM–99, Mathematics and Computer Science Division, Argonne National Laboratory, September 1987.

[10] B. Garbow, J. Boyle, J. Dongarra, and C. Moler. *Matrix Eigensystem Routines – EISPACK Guide Extension*, volume 51 of *Lecture Notes in Computer Science*. Springer Verlag, New York, 1977.

[11] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1983.

[12] Linda Kaufman. Banded eigenvalue solvers on vector machines. *ACM Transactions on Mathematical Software*, 10(1):73–86, March 1984.

[13] Bruno Lang. *Parallele Reduktion symmetrischer Bandmatrizen auf Tridiagonalgestalt*. PhD thesis, Universität Karlsruhe (TH), 1991.

[14] Bruno Lang. Reducing symmetric banded matrices to tridiagonal form – a comparison of a new parallel algorithm with two serial algorithms on the iPSC/860. Technical report, Universität Karlsruhe, Institut für Angewandte Mathematik, January 1992.

[15] H. Rutishauser. On Jacobi rotation patterns. In *Proc. of Symposia in Applied Mathematics, Vol. 15, Experimental Arithmetic, High Speed Computing and Mathematics*, pages 219–239, 1963.

[16] Robert Schreiber and Charles Van Loan. A storage efficient WY representation for products of Householder transformations. *SIAM Journal on Scientific and Statistical Computing*, 10(1):53–57, 1989.

[17] Hans Rudolph Schwarz. Tridiagonalization of a symmetric band matrix. *Numerische Mathematik*, 12:231–241, 1968.

[18] B. Smith, J. Boyle, J. Dongarra, B. Garbow, Y. Ikebe, V. Klema, and C. B. Moler. *Matrix Eigensystem Routines – EISPACK Guide*. Springer-Verlag, New York, second edition, 1976.