Table 4: Timings for Wilkinson Shift and Perfect Shift Algorithms

Case	Routine	$n = 100$ time (secs.)	$n = 100$ no. vector iterations	$n = 100$ no. inner loop passes	$n = 200$ time (secs.)	$n = 200$ no. vector iterations	$n = 200$ no. inner loop passes
Case 1. Constant Diagonal	SSTEQR	7.6	176	8970	81	335	33741
	SSTEPS	6.8	141	7338	72	292	30107
Case 2. Graded from Small to Large	SSTEQR	31	113	6776	200	240	27794
	SSTEPS	31	113	6776	200	242	27858
Case 3. Graded from Large to Small	SSTEQR	31	113	6776	200	240	27794
	SSTEPS	31	113	6776	200	242	27858
Case 4. Sawtooth	SSTEQR	9.8	115	2126	39	214	4289
	SSTEPS	9.3	105	2011	45	200	4084
Case 5. More Diagonally Dominant Sawtooth	SSTEQR	8.9	100	1397	20	184	2746
	SSTEPS	8.2	96	1426	20	190	2762
Case 6. Deflatable Sawtooth	SSTEQR	.69	100	755	2.6	199	1535
	SSTEPS	.72	91	720	2.6	183	1471
Case 7. Random	SSTEQR	11	178	9723	150	358	37207
	SSTEPS	9.3	150	8200	150	314	31379
Case 8. Random	SSTEQR	7.9	186	9307	110	361	35706
	SSTEPS	7.4	155	7941	120	309	31721

Case 4. Sawtooth	IMTQL1	.21	.45	.97
	SSTEQR	.11	.27	.82
	TQLRAT	.12	.28	1.3
	SSTERF	.12	.34	1.5
Case 5. More Diagonally Dominant Sawtooth	IMTQL1	.12	.26	.72
	SSTEQR	.09	.20	.65
	TQLRAT	.07	.15	.63
	SSTERF	.08	.19	.59
Case 6. Deflatable Sawtooth	IMTQL1	.08	.18	.54
	SSTEQR	.05	.13	.53
	TQLRAT	.04	.18	.62
	SSTERF	.04	.11	.44
Case 7. Random	IMTQL1	.89	2.9	18.
	SSTEQR	.52	1.9	11.
	TQLRAT	.34	1.2	8.0
	SSTERF	.45	1.5	8.9
Case 8. Random	IMTQL1	.91	3.0	18.
	SSTEQR	.51	1.9	12.
	TQLRAT	.32	1.2	8.9
	SSTERF	.42	1.5	11.

Table 2: Accuracy of Different QR/QL Strategies. ($\epsilon = .596\epsilon - 07$)

Case	Routine	Max. Abs. Error over Matrix Norm	Max. Rel. Error
Case 2. Graded from Small to Large	IMTQL1	.77e-07	.20e+02
	SSTEQR(1)	.14e-06	.12e-02
	SSTEQR(2)	.14e-06	.12e-02
	SSTEQR(3)	.14e-06	.12e-02
	TQLRAT	.14e-06	.84e-06
	SSTERF(1)	.21e-06	.19e-05
	SSTERF(2)	.21e-06	.19e-05
	SSTERF(3)	.21e-06	.19e-05
Case 3. Graded from Large to Small	IMTQL1	.70e-07	.12e-02
	SSTEQR(1)	.77e-07	.27e-02
	SSTEQR(2)	.14e-06	.12e-02
	SSTEQR(3)	.14e-06	.12e-02
	TQLRAT	.11e-06	.55e+06
	SSTERF(1)	.79e-07	.69e+06
	SSTERF(2)	.21e-06	.19e-05
	SSTERF(3)	.21e-06	.19e-05

Table 3: Timings for Implicit QR/QL and Root-Free Variants

Case	Routine	Time in seconds		
		n=100	n=200	n=500
Case 1. Constant Diagonal	IMTQL1	.86	3.1	19.
	SSTEQR	.50	1.7	10.
	TQLRAT	.32	1.2	7.8
	SSTERF	.38	1.4	8.7
Case 2. Graded from Small to Large	IMTQL1	failed		
	SSTEQR	.30	1.2	7.7
	TQLRAT	.18	.74	4.2
	SSTERF	.19	.74	4.4
Case 3. Graded from Large to Small	IMTQL1	.70	2.9	14.
	SSTEQR	.34	1.4	7.6
	TQLRAT	.19	.63	3.6
	SSTERF	.20	.90	4.4

continued from previous page

Case	Routine	No. QR/QL Iterations	No. Passes through Inner Loop	QR or QL or No. Switches
Case 5. More Diagonally Dominant Sawtooth	IMTQL1	136	1429	QL
	SSTEQR(1)	100	1397	QL
	SSTEQR(2)	106	1501	10
	SSTEQR(3)	100	1397	QL
	TQLRAT	89	1037	QL
	SSTERF(1)	82	1325	QL
	SSTERF(2)	82	1311	16
	SSTERF(3)	82	1325	QL
Case 6. Deflatable Sawtooth	IMTQL1	141	912	QL
	SSTEQR(1)	132	903	QL
	SSTEQR(2)	124	865	7
	SSTEQR(3)	100	755	8
	TQLRAT	87	488	QL
	SSTERF(1)	111	772	QL
	SSTERF(2)	99	705	7
	SSTERF(3)	59	496	8
Case 7. Random	IMTQL1	199	10455	QL
	SSTEQR(1)	184	9716	QL
	SSTEQR(2)	177	9798	4
	SSTEQR(3)	178	9723	QR
	TQLRAT	193	10033	QL
	SSTERF(1)	182	9714	QL
	SSTERF(2)	176	9866	4
	SSTERF(3)	173	9514	QR
Case 8. Random	IMTQL1	201	10278	QL
	SSTEQR(1)	186	9307	QL
	SSTEQR(2)	187	9354	4
	SSTEQR(3)	186	9307	QL
	TQLRAT	193	9579	QL
	SSTERF(1)	183	9213	QL
	SSTERF(2)	184	9155	4
	SSTERF(3)	183	9213	QL

Table 1: Iteration Counts for Different QR/QL Strategies

Case	Routine	No. QR/QL Iterations	No. Passes through Inner Loop	QR or QL or No. Switches
Case 1. Constant Diagonal	IMTQL1	195	10520	QL
	SSTEQR(1)	176	8970	QL
	SSTEQR(2)	172	8963	1
	SSTEQR(3)	176	8970	QL
	TQLRAT	184	9905	QL
	SSTERF(1)	170	8926	QL
	SSTERF(2)	170	8926	QL
	SSTERF(3)	170	8926	QL
Case 2. Graded from Small to Large	IMTQL1	failed		
	SSTEQR(1)	113	6776	QL
	SSTEQR(2)	113	6776	QL
	SSTEQR(3)	113	6776	QL
	TQLRAT	57	4254	QL
	SSTERF(1)	57	4189	QL
	SSTERF(2)	57	4189	QL
	SSTERF(3)	57	4189	QL
Case 3. Graded from Large to Small	IMTQL1	199	9999	QL
	SSTEQR(1)	197	9997	QL
	SSTEQR(2)	113	6776	QR
	SSTEQR(3)	113	6776	QR
	TQLRAT	135	3905	QL
	SSTERF(1)	184	9955	QL
	SSTERF(2)	57	4189	QR
	SSTERF(3)	57	4189	QR
Case 4. Sawtooth	IMTQL1	147	2391	QL
	SSTEQR(1)	115	2126	QL
	SSTEQR(2)	130	2502	15
	SSTEQR(3)	115	2126	QL
	TQLRAT	93	2664	QL
	SSTERF(1)	75	2417	QL
	SSTERF(2)	75	2417	1
	SSTERF(3)	75	2417	QL

continued on next page

References

1 J. Demmel and W. Kahan. *Computing Small Singular Values of Bidiagonal Matrices with Guaranteed High Relative Accuracy.* LAPACK Working Note No. 3, Argonne National Laboratory technical report MCS-TM-110, February 1988.

2 B. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, N.J., 1980.

3 C. Reinsch. *A Stable Rational QR Algorithm for the Computation of the Eigenvalues of an Hermitian, Tridiagonal Matrix.* Num. Math. 25:591-597, 1971.

values. It is important to understand how this affects the performance. We know that if an intermediate subdiagonal element were zero, it would remain zero in all iterations. In such a case if λ were an eigenvalue belonging to the upper tridiagonal matrix, we could never make the eigenvalue move down to the bottom right-hand corner even with an exact λ and exact computation. One might expect then that if a subdiagonal is small, it might take several iterations to make an eigenvalue that belonged to the upper tridiagonal matrix move down to the bottom if we were using a λ that was not exact. In fact, the situation is worse; a succession of smallish (but by no means pathologically small) subdiagonal elements has the effect of almost decoupling the top tridiagonal matrix from the bottom and several iterations may be required even if λ is a correctly rounded eigenvalue. The phenomenon has little to do with the rounding errors made in the QR algorithm, and this actually plays only a minimal role in the performance. With an accurate but non-exact λ it may take a number of steps of QR executed in exact arithmetic to move the relevant eigenvalue to the bottom.

The usual Wilkinson shift strategy (SSTEQR) was compared with the perfect shift algorithm (SSTEPS), and results are shown in Table 4. The table gives times on the Sun 4 for problems of size $n = 100$ and $n = 200$, as well as counts of the number of *vector* iterations and the number of passes through the inner loop during the vector iterations. This latter number should be roughly proportional to the total work. Similar experiments were run on a CRAY X-MP with the same general behavior.

While the perfect shift strategy gives some improvement over the standard Wilkinson shift for most problems, the difference is not great. On a few problems, it is slightly slower. It is feared that when eigenvalues are not computed to high relative precision, the computed eigenvalues might actually be ineffective shifts. For this reason, it was decided to use the standard Wilkinson shift algorithm in the LAPACK routine SSTEQR.

$$TQLRAT: \sim 3.9E-5 \quad seconds$$

$$SSTERF: \sim 4.7E-5 \quad seconds$$

From these timings it appears that the root-free algorithms offer significant advantage over the standard implicit QL method on a machine with no hardware square root. The Reinsch algorithm TQLRAT is somewhat faster than the Pal-Walker-Kahan variant SSTERF, but because the difference is not great, we opted to use the more elegant Pal-Walker-Kahan algorithm in LAPACK.

7 Wilkinson's Shift vs. Perfect Shifts for Eigenvector Computation

When it is desired to compute the eigenvectors of the tridiagonal matrix (or of a full matrix that has been reduced to tridiagonal form) as well as the eigenvalues, one might consider the following strategy:

1. Make a copy T' of the matrix T.
2. Compute an eigenvalue of T' using QR/QL with the standard Wilkinson shift.
3. Use this eigenvalue as a shift in a QR/QL eigenvector computation with T. After finding an eigenvector (probably, but not necessarily, the one corresponding to the computed eigenvalue), go to step 1.

When the eigenvectors are required, the bulk of the work is in computing the product of the plane rotations. The following observation may be used to reduce the amount of work. Suppose we had an exact eigenvalue, λ, and we were to perform one exact step of the QR algorithm using λ as a shift. For this one step the eigenvalue would be moved down to the bottom right hand corner, and the final subdiagonal element would be zero. Hence, the eigenvector of the transformed tridiagonal matrix corresponding to λ would be e_n (the last column of the identity matrix) and the corresponding eigenvector of the original tridiagonal matrix would be the product of the successive plane rotations.

A good algorithm will produce very accurate eigenvalues, but they are not in general exact. Since they are usually irrational we cannot have exact

6 Root-Free vs. Standard Methods

Standard versions of the QR/QL algorithm require computing square roots in the inner loop. On machines that do not have a hardware square root function, this computation can be time-consuming. Considerable effort has gone into deriving algorithms that are mathematically equivalent to QR/QL but do not require square roots.

One such algorithm, derived by Reinsch [3], has been implemented in the EISPACK routine TQLRAT. At one point in the Reinsch algorithm it is necessary to divide by a quantity that could be zero. If it is too small, it is replaced by something on the order of the machine precision. This procedure can be justified in a backward error sense, as changing the problem by an amount that is tiny compared to the norm of the matrix; but if this replacement is made, then very small eigenvalues will certainly not be computed accurately in a relative sense.

A root-free algorithm that avoids this check for a zero divisor was developed by Pal, Walker, and Kahan [2]. It, too, may fail to find very small eigenvalues to high relative accuracy, but it is somewhat more elegant than the Reinsch algorithm in that there are no requirements for replacing tiny divisors. This algorithm has been implemented in the LAPACK routine SSTERF.

Table 3 shows timings on the Sun-4 for EISPACK routines IMTQL1 and TQLRAT and for LAPACK routines SSTEQR and SSTERF (using strategy (3) for selecting between QR and QL iteration). The Sun-4 does not have a hardware square root function, and, in fact, even if it did, it would not help EISPACK routine IMTQL1 as it stands. This routine was written to use a software square root routine, PYTHAG, which should certainly be replaced if a better square root function is available. (We did not modify the EISPACK routines for this test, except for the convergence test, described earlier.) Test problems of size $n = 100$, 200, and 500 were used. ¿From the timings in the $n = 100$ column of Table 3 and the counts of passes through the inner loop in Table 1, one can determine the approximate time required for a single pass through the inner loop in each of these algorithms:

$$IMTQL1: \ \sim 8.4E-5 \ \ seconds$$

$$SSTEQR: \ \sim 5.1E-5 \ \ seconds$$

of number of iterations. Surprisingly, TQLRAT (which uses QL iteration) actually required the fewest trips through the inner loop and, as will be seen in the next section, the least amount of time. For a fixed algorithm, however, the QR variant appears to offer a significant advantage in both number of iterations and total work. With either option (2) or option (3), the entire computation happened to be performed with QR iteration, resulting in significantly fewer iterations and passes through the inner loop than the corresponding QL methods.

In problem 6, the deflatable sawtooth pattern, the advantages of option (3) over either option (2) or option (1) are seen. Strategy (2) – comparing the top and bottom diagonal elements to decide between QR or QL iteration – is really inappropriate here since the matrix splits and only one block of the matrix is operated on at a time. Strategy (2) switched 7 times between QR and QL iteration, but still required more iterations and more passes through the inner loop than strategy (3). A more sophisticated strategy allowing switching from QR to QL within blocks, always comparing the top and bottom diagonal elements of the block that is actually being worked on, might offer some advantage over strategy (3) but this makes bookkeeping (keeping track of which parts of the matrix have and have not been reduced to diagonal form) more difficult.

For most problems, each of these routines gave similar levels of accuracy, computing eigenvalues to high absolute precision (compared to the norm of the matrix), but not necessarily to high relative precision. Exceptions were problems 2 and 3, which were strictly graded in one direction or the other. For these problems, the root-free variants were able to obtain nearly full relative accuracy, provided the proper choice was made between QR and QL iteration. Results for these two problems are shown in Table 2.

Based on these results, it was decided to use strategy (3) for choosing between QL and QR iteration in the LAPACK routines SSTEQR and SSTERF. This is the version that will be used in all following comparisons.

smaller, compute the next eigenvalue using QL iteration; if the bottom element is smaller, compute the next eigenvalue using QR iteration.

3. At the beginning, determine whether the matrix splits; compare the top diagonal element to the last diagonal element before the split, or the bottom diagonal element if the matrix does not split. Choose QL or QR iteration for that part of the matrix according to whether the top or bottom element is smaller. If the matrix does split, then when that block is completed, find where the matrix next splits and use the same criterion for applying QL or QR iteration to this block. Continue.

Iteration counts for each version, along with iteration counts for the EISPACK QL routines, for problems of size $n = 100$, in single precision, are listed in Table 1. Both the number of QR/QL iterations and the total number of passes through the inner loop for each algorithm are listed. Note that QR/QL iterations that occur early in the computation, before many eigenvalues have converged or before the matrix has split, require more passes through the inner loop and hence more work than the later iterations. The total work is roughly proportional to the number of passes through the inner loop, but may not be proportional to the number of iterations. In this and all other comparisons with EISPACK, the only modification that has been made to the EISPACK routines is in their convergence tests. All routines use the criterion

$$|e(m)| \leq \epsilon \cdot (|d(m)| + |d(m+1)|) \quad \textit{for QL iteration, or}$$
$$|e(m-1)| \leq \epsilon \cdot (|d(m)| + |d(m-1)|) \quad \textit{for QR iteration}$$

to determine whether $d(m)$ has converged to an eigenvalue.

Note that the EISPACK routine IMTQL1 failed on problem 2. This is because more than 30 iterations were required to find the first eigenvalue, and EISPACK allows no more than 30 iterations for any one eigenvalue. Once the first eigenvalue has been found, however, each of the remaining eigenvalues can be computed in far fewer than 30 iterations. Hence the LAPACK routines were written to allow a total of no more than $30n$ iterations for all eigenvalues, but not to restrict the number of iterations for any one eigenvalue.

In problem 3, which is graded from large to small, the methods that allow for QR iteration significantly outperform those that do not, in terms

be expected to give high relative accuracy. We will be satisfied if the absolute errors divided by the norm of the matrix are less than the problem size n times the machine precision ϵ. Eigenvectors will be checked by computing the residual norm, $||Tv - \lambda v||$, and seeing that this is less than ϵ times the norm of T. Results reported here are for the single-precision routines, though similar experiments have been performed with the double-precision versions. Tests were run on a Sun-4, where the machine precision ϵ is $2^{-24} \approx .596e-07$.

4 Timing Tests and Operation Counts

Timings reported here were obtained on the Sun-4. Where possible, they are accompanied by operation counts or iteration counts, which are less dependent on the particular machine used. In most cases, decisions about which method to use are based on such work estimates rather than on actual timings.

5 QR vs. QL Iterations.

If the smaller elements of the tridiagonal matrix are located at the top, one might expect faster convergence using the QL method to annihilate these small elements first; conversely, if the smaller elements are at the bottom, then the QR iteration might be expected to converge faster. If the relative sizes of the top and bottom elements change during the course of the computation, one might expect to gain some advantage by switching between QL and QR iterations.

The EISPACK routines IMTQL1 and TQLRAT use QL iteration only. IMTQL1 uses the implicit QL algorithm while TQLRAT uses a root-free variant developed by Reinsch [3]. LAPACK routine SSTEQR has been written based on the same implicit algorithm as IMTQL1. LAPACK routine SSTERF is based on a root-free algorithm developed by Pal, Walker, and Kahan [2], which will be described further in the following section. The LAPACK routines were tested with three variations:

1. Use QL iteration only.

2. At the beginning, and after each eigenvalue converges, check whether the top or bottom diagonal element is smaller. If the top element is

$$D_{ii} \;=\; 10^{11 \;-\; ((i-1) \bmod 12)}, \quad if\ [\frac{i-1}{12}]\ is\ odd.$$

Test problem 5, labelled **"More Diagonally Dominant Sawtooth,"** is obtained from problem 4 by multiplying each diagonal element of T_4 by 10:

$$T_5 \;=\; T_4 \quad except\ that\ \ T_5(i,i) = 10 \cdot T_4(i,i), \quad i = 1, ..., n.$$

Test problem 6, labelled **"Deflatable Sawtooth,"** is obtained from problem 4 by setting every twelfth subdiagonal (and superdiagonal) element to zero:

$$T_6 \;=\; T_4 \quad except\ that\ \ T_6(12i, 12i+1) = T_6(12i+1, 12i) = 0, \quad i = 1, 2, ...$$

Test problems 7 and 8, labelled **"Random,"** are symmetric tridiagonal matrices whose entries are uniformly distributed random numbers between -1 and 1. The results obtained on these two problems are representative of those obtained on a larger sample of random matrices of this form.

3 Accuracy Tests

Eigenvalues computed by the different routines are compared with the squared singular values returned by the LAPACK routine DBDSQR [1]. This routine is able to compute singular values of a bidiagonal matrix to high relative accuracy, so that even if the singular values range over many orders of magnitude in size (as they do in many of these test problems), the smallest as well as the largest will be computed accurately. The tridiagonal matrices in test problems 1 through 6 are positive definite, so their bidiagonal Cholesky factors can be computed, and the squared singular values of the computed Cholesky factors will also approximate the eigenvalues of the original matrix to high relative accuracy [1]. In problems 7 and 8, a multiple of the identity matrix was added to each tridiagonal to make it diagonally dominant before performing its Cholesky factorization. This multiple was then subtracted from the values returned by DBDSQR. In these problems the eigenvalues did not range over many orders of magnitude, so no special procedure was needed to obtain high relative accuracy.

Both relative and absolute errors in the computed eigenvalues will be reported, but, as will be seen, none of the QR/QL methods considered here can

computation. Based on these results, LAPACK routines have been developed to incorporate the most promising variations. The new routines are compared with similar EISPACK routines.

2 Test Problems

The test problems include a variety of symmetric tridiagonal matrices, some of which are strongly graded, with diagonal elements ranging over twelve orders of magnitude, in increasing order, in decreasing order, or in a "sawtooth" pattern alternating between increasing and decreasing. Test problem 1, labelled **"Constant Diagonal,"** is the matrix

$$T_1 \;=\; tridi(-1, 2, -1).$$

Test problem 2, labelled **"Graded from Small to Large,"** is obtained by multiplying this matrix on the left and right by the square root of the diagonal matrix whose elements vary from 1 to 10^{12} in a geometric progression:

$$T_2 \;=\; D^{1/2}\, T_1\, D^{1/2}, \quad where \;\; D_{ii} = d^{i-1}, \;\; d = 10^{12/(n-1)}, \;\; i = 1, ..., n.$$

Test problem 3, labelled **"Graded from Large to Small,"** is obtained by multiplying T_1 on the left and right by the square root of the diagonal matrix whose elements vary from 10^{12} down to 1 in a geometric progression:

$$T_3 \;=\; D^{1/2}\, T_1\, D^{1/2}, \quad where \;\; D_{ii} = d^{n-i}, \;\; d = 10^{12/(n-1)}, \;\; i = 1, ..., n.$$

Test problem 4, labelled **"Sawtooth,"** is again obtained from problem 1 by multiplying on the left and right by the square root of a diagonal matrix. The diagonal elements go from 1 up to 10^{11} by factors of 10 and then from 10^{11} back down to 1 by factors of 1/10. This pattern is repeated until the end of the matrix is reached:

$$T_4 \;=\; D^{1/2}\, T_1\, D^{1/2}, \quad where$$
$$D_{ii} = 10^{(i-1) \bmod 12}, \quad if\ [\frac{i-1}{12}]\ is\ even,$$

Experiments with QR/QL Methods for the Symmetric Tridiagonal Eigenproblem

A. Greenbaum and J. Dongarra*

November 21, 1989

Abstract

Numerical experiments used in determining which variants of the QR/QL algorithm to include in LAPACK are described. Timing and accuracy comparisons are presented for the different methods applied to the symmetric tridiagonal eigenproblem. Specifically, comparisons are made between root-free and standard versions, between QL and QR iterations and dynamic strategies for switching between the two, and between Wilkinson's shift and the perfect shift strategy for the eigenvector computation. LAPACK routines that incorporate the most promising of these strategies are then compared with the corresponding EISPACK routines.

1 Introduction

Numerical experiments with variants of the QR/QL algorithm for the symmetric tridiagonal eigenproblem are reported. Accuracy and timing comparisons are made between root-free and standard versions, between QR and QL iterations and strategies for dynamically switching between the two, and between Wilkinson's shift and the perfect shift strategy for the eigenvector

*This work was supported by NSF Grant No. ASC-8715728.