A Stable, Rational QR Algorithm for the Computation of the Eigenvalues of an Hermitian, Tridiagonal Matrix

By Christian H. Reinsch

Abstract. The most efficient program for finding all the eigenvalues of a symmetric matrix is a combination of the Householder tridiagonalization and the QR algorithm. The latter, if carried out in a natural way, requires \(4n\) additions, \(10n\) multiplications, \(2n\) divisions, and \(n\) square roots per iteration (\(n\) the order of the matrix). In 1963, Ortega and Kaiser showed that the process can be carried out using no square roots (and saving \(7n\) multiplications). However, their algorithm is unstable and several modifications were suggested to increase its accuracy. We, too, want to give such a modification together with some examples demonstrating the achieved accuracy.

1. Introduction. In 1961 Francis [4] proposed the QR transformation, an offspring of Rutishauser's LR transformation [8], for the computation of the eigenvalues of a general matrix. He considered his method to be inefficient for Hermitian matrices but, fortunately, it soon turned out that, contrary to his original opinion, the method is especially efficient for this class of matrices, provided the given matrix is first reduced by Householder's method to real tridiagonal form and provided that shifts are used to accelerate the rate of convergence. (A description of this technique can be found in [10], for tested ALGOL programs see [6], [3], [2], the properties of the now generally adopted shift are described in [11].)

Ortega and Kaiser [7] pointed out that by avoiding square roots the efficiency of this algorithm can be further increased (though if all eigenvalues are to be computed, it is already superior to all other known methods). The algorithm which they proposed, however, was unstable and several modifications were suggested (e.g., [9], [5] and others, not published). We, too, want to give such a modification here, together with some examples demonstrating the achieved accuracy.

2. The Algorithm. Let \(A\) be the shifted matrix with diagonal entries \(a_1, \ldots, a_n\) and subdiagonal entries \(b_1, \ldots, b_{n-1}\). As is well known, a QR step consists of the orthogonal-triangular decomposition, \(A = QR\), and the recombination in reversed order, \(A = RQ\). For tridiagonal matrices, the decomposition is usually done by the application of \(n - 1\) plane rotations \(P_1^T, \ldots, P_{n-1}^T\) from the left to \(A\) to produce the upper triangular matrix \(R\) with diagonal \(r_1, \ldots, r_n\), first superdiagonal \(q_1, \ldots, q_{n-1}\),...
and second superdiagonal \( t_1, \cdots, t_{n-2} \). If

\[
P_i^T = \begin{bmatrix} 1 & & & & \\ & c_i & s_i & & \\ & & -s_i & c_i & \\ & & & & \ddots \end{bmatrix}, \quad P_{j-1}^T P_j^T \cdots P_1^T A = \begin{bmatrix} r_{i-1} & q_{i-1} & t_{i-1} \\ p_i & b_i c_{i-1} \\ b_i & a_{i+1} & b_{i+1} \end{bmatrix},
\]

then it follows by induction that

\[ p_1 = a_1, \quad c_0 = 1; \]

\[ r_i = (p_i^2 + b_i^2)^{1/2}, \] \[ c_i = p_i / r_i, \]

\[ s_i = b_i / r_i, \quad j = 1, \cdots, n - 1; \]

\[ a_i = c_i c_{i-1} b_i + s_i a_{i+1}, \]

\[ p_{i+1} = c_i a_{i+1} - s_i c_{i-1} b_i, \]

\[ c_{n-1} = p_n. \]

For the recombination, we have in the tridiagonal case \( \bar{A} = RP_1 \cdots P_{n-1} \) giving

\[ \bar{a}_i = r_i c_{i-1} c_i + q_i s_i, \]

\[ = p_i c_{i-1} + q_i s_i, \quad j = 1, \cdots, n - 1; \]

\[ b_i = r_{i+1} s_i, \]

\[ \bar{a}_n = r_n c_{n-1}. \]

Thus, \( 4n \) additions, \( 10n \) multiplications, \( 2n \) divisions, and \( n \) square roots are necessary per iteration. This algorithm is realized in the programs of [2]. To avoid square roots, Ortega and Kaiser introduced the quantities

\[ h_i = p_i c_{i-1} \]

(which they call \( \gamma_i \)) and computed \( p_i \) from \( h_i^2 / (1 - s_{i-1}^2) \) which is obviously inaccurate if \( s_{i-1}^2 \) is near to unity. We prefer to compute a quantity \( g_i \) defined by

\[ g_i = p_i / c_{i-1} \]

and to compute \( h_i \) from it. From the last equation of (2) and from \( c_i b_i = s_i p_i \) one obtains the recurrence relation

\[ g_{i+1} = a_{i+1} - s_i c_{i-1} b_i / c_i = a_{i+1} - s_i c_{i-1} b_i^2 / s_i p_i = a_{i+1} - b_i^2 / g_i, \]

while according to Ortega and Kaiser, the second equation of (4) can be transformed into

\[ a_i = h_i + c_i c_{i-1} b_i s_i + s_i a_{i+1} s_i = h_i + s_{i-1}^2 (h_i + a_{i+1}). \]
Thus, the algorithm without square roots becomes

\[
\begin{align*}
&g_1 = h_1 = a_1, \quad s_0^2 = 0; \\
p_i^2 = g_i h_i, \\
r_i^2 = p_i^2 + b_i^2, \\
b^2_{i-1} = r_i^2 s_{i-1}^2 \quad (j \neq 1), \\
n_j^2 = r_j^2 / r_i^2, \\
\tilde{a}_i = h_i + s_i^2 (h_i + a_{i+1}), \\
g_{i+1} = a_{i+1} - b_i^2 / g_i, \\
h_{i+1} = g_{i+1} p_i^2 / r_i^2, \\
\tilde{a}_n = h_n, \quad \tilde{b}_{n-1} = g_n h_n s_{n-1}^2.
\end{align*}
\]

(8)

Note that the squares of the subdiagonal elements rather than the elements themselves are the given data. This algorithm needs \(4n\) additions, \(4n\) multiplications, and \(3n\) divisions (\(n\) additions less than Ortega and Kaiser but \(n\) multiplications and \(n\) divisions more). The iteration is repeated until the last off-diagonal entry becomes smaller than a given tolerance \(\Delta\).

It should be mentioned that the \(g_i\) are the well-known quotients of consecutive principal minors of the shifted matrix \(A\), which are also used in the bisection process [1]. This can be used to advantage to assign ordinals to computed eigenvalues. As is the case there, a vanishing \(g_j\) has to be replaced by a small nonzero value \(\delta\) equivalent to a perturbation of the diagonal entry \(a_j\). \(\delta\) has to be chosen smaller than \(2\Delta\) in order to avoid indefinite cycling: if \(g_j\) is replaced by \(\delta\) then \(\tilde{b}_{n-1} = \delta^2 s_{n-1}^2 c_{n-1}^2 \leq \delta^2 / 4 < \Delta^2\), and the iteration terminates. In any case, the computed values of \(g_j (j = 1, \ldots, n)\) are always the exact values corresponding to slightly modified entries of the matrix \(A\). Rounding errors in the evaluation of the remaining expressions are obviously harmless. For technical reasons, a decomposition starting with the lower end of the tridiagonal matrix is preferable (called the "\(QL\) algorithm"). This is merely achieved by the replacement

\[
\begin{align*}
d_n, \ldots, d_1 & \quad \text{for } a_1, \ldots, a_n, \\
e_{n-1}^2, \ldots, e_1^2 & \quad \text{for } b_1^2, \ldots, b_{n-1}^2,
\end{align*}
\]

(and similarly for the entries of \(A\)). Introducing the fake quantity \(e_n^2\) we obtain the procedure (without shift):

\[
\begin{align*}
g : = h : = d_n; \quad s^2 : = 0; \\
i : = n - 1(-1)^1 : \\
p^2 : = g \times h; \quad r^2 : = p^2 + e_i^2; \\
\tilde{e}_i^2 : = s^2 \times r^2; \quad \tilde{s}^2 : = e_i^2 / r^2; \\
\tilde{d}_{i+1} : = h + s^2 \times (h + d_i); \\
g : = d_i - e_i^2 / g; \quad h : = g \times p^2 / r^2; \\
\tilde{d}_i : = h; \quad \tilde{e}_i^2 : = g \times h \times s^2.
\end{align*}
\]

(9)
The computation of the shift, the tests for splitting and convergence can be done as in the usual QR algorithm (see [2]).

3. Test Examples. The algorithm (9) was embedded in the organizational scheme of procedure TQL1 [2]. The following numerical results were obtained on the AEG-TELEFUNKEN computer TR-4 of the Leibnitz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, München, with machine precision $2^{-35} \approx 2.91 \times 10^{-15}$. Two consecutive machine numbers in the interval [1, 16) have a distance $2^{-34} \approx 5.82 \times 10^{-14}$, and this would be the appropriate unit to measure most of the errors $\tilde{\lambda}_i - \lambda_i$ listed below. Rounded decimal equivalents are listed in sequence as they were computed.

First Example.

\[
\begin{array}{c}
\text{order:} 5, \\
\text{diagonal:} (0, 0, 0, 0, 0), \\
\text{subdiagonal:} (1, 1, 1, 1).
\end{array}
\]

This example served as a test for formal correctness. In the first iteration $g_2$ and $g_5$ become zero for the chosen shift and have to be replaced by a small tolerance quantity.

\[
\begin{array}{cccc}
\text{Computed eigenvalues } \tilde{\lambda}_i & \tilde{\lambda}_i - \lambda_i & \text{Iterations} \\
-0.99999999999996 & 0.4 & 1 \\
1.00000000000000 & 0 & 1 \\
-5.1410 \times 10^{-11} & -0.5 \times 10^{-11} & 4 \\
1.732050 8077 & 11.2 & 1 \\
-1.732050 8075 & 6.3 & 0
\end{array}
\]

\[7\]

Second Example. Wilkinson's matrix $W_{21}$,

\[
\begin{array}{c}
\text{order:} 21, \\
\text{diagonal:} (10, 9, \ldots, -9, -10), \\
\text{subdiagonal:} (1, \ldots, 1).
\end{array}
\]
This example was chosen since the exact eigenvalues were known to 24 decimal places.

<table>
<thead>
<tr>
<th>Computed eigenvalues $\hat{\lambda}_i$</th>
<th>$\hat{\lambda}_i - \lambda_i$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.74619 4183</td>
<td>-0.8</td>
<td>3</td>
</tr>
<tr>
<td>9.21067 86472</td>
<td>-11.9</td>
<td>2</td>
</tr>
<tr>
<td>8.03894 11193</td>
<td>-3.3</td>
<td>2</td>
</tr>
<tr>
<td>7.00395 20027</td>
<td>1.1</td>
<td>2</td>
</tr>
<tr>
<td>6.00022 56802</td>
<td>5.5</td>
<td>2</td>
</tr>
<tr>
<td>5.00000 81589</td>
<td>17.9</td>
<td>2</td>
</tr>
<tr>
<td>4.00000 02049</td>
<td>-12.1</td>
<td>2</td>
</tr>
<tr>
<td>3.00000 00037</td>
<td>-8.3</td>
<td>2</td>
</tr>
<tr>
<td>2.00000 00001</td>
<td>0.4</td>
<td>2</td>
</tr>
<tr>
<td>0.99999 999999</td>
<td>-1.2</td>
<td>2</td>
</tr>
<tr>
<td>$4.014_{10^{-11}}$</td>
<td>$4.0 \times 10^{-11}$</td>
<td></td>
</tr>
<tr>
<td>-1.00000 00005</td>
<td>-46.5</td>
<td>1</td>
</tr>
<tr>
<td>-2.00000 00001</td>
<td>-6.2</td>
<td>2</td>
</tr>
<tr>
<td>-3.00000 00040</td>
<td>-15.0</td>
<td>1</td>
</tr>
<tr>
<td>-4.00000 02057</td>
<td>-6.3</td>
<td>2</td>
</tr>
<tr>
<td>-5.00000 81591</td>
<td>-4.1</td>
<td>1</td>
</tr>
<tr>
<td>-6.00022 56806</td>
<td>-4.6</td>
<td>2</td>
</tr>
<tr>
<td>-7.00395 20030</td>
<td>$-3.6 \times 10^{-10}$</td>
<td>1</td>
</tr>
<tr>
<td>-8.03894 11197</td>
<td>-4.3</td>
<td>1</td>
</tr>
<tr>
<td>-9.21067 86491</td>
<td>-17.4</td>
<td>1</td>
</tr>
<tr>
<td>-10.74619 4185</td>
<td>-20.9</td>
<td>0</td>
</tr>
</tbody>
</table>

Third Example. Wilkinson's matrix $W_{21}^+$,

order: 21,
diagonal: (10, 9, ⋮, 0, ⋮, 9, 10),
subdiagonal: (1, ⋮, 1).

Here, too, were the eigenvalues known to 24 decimals. The matrix has a number of close pairs of eigenvalues, and earlier algorithms for the rational $QR$ transformation gave only poor results.
Composed eigenvalues $\lambda_i$ | $\lambda_i - \lambda$, | Iterations
--- | --- | ---
10.74619 4183 | -0.8 | 3
10.74619 4183 | 5.0 | 2
9.21067 86473 | -3.3 | 2
9.21067 86473 | -8.9 | 1
8.03894 11157 | -9.1 | 3
8.03894 11228 | -6.3 | 0
7.00395 17986 | -1.6 | 3
7.00395 22095 | -4.0 | 0
6.00021 75223 | -0.5 | 3
6.00023 40316 | 0.0 | 0
5.00024 44249 | -9.1 $\times 10^{-11}$ | 3
4.99978 24777 | -6.3 | 1
4.00435 40235 | 10.8 | 3
3.99604 82015 | 13.2 | 1
3.04309 92925 | -8.3 | 3
2.96105 88842 | -2.5 | 1
2.13020 92192 | -18.6 | 2
1.78932 13524 | -31.7 | 1
0.94753 436752 | -0.7 | 2
0.25380 581678 | -31.9 | 1
-1.12544 15223 | -21.8 | 0
35

Fourth Example.

order: 21,
diagonal: (0, 0, 0, 0, 5, 5, ..., 5, 5, 0, 0, 0),
subdiagonal: (1, ..., 1).

This matrix has also several close pairs of eigenvalues. The same method in double precision was used to compute them in sufficient accuracy to permit a reliable computation of $\lambda_i - \lambda_i$. 
### Computed eigenvalues $\lambda_i$ and iterations

<table>
<thead>
<tr>
<th>$\lambda_i$</th>
<th>$\lambda_i - \lambda_j$</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.68610</td>
<td>208438</td>
<td></td>
</tr>
<tr>
<td>0.68610</td>
<td>208455</td>
<td></td>
</tr>
<tr>
<td>-1.64133</td>
<td>43818</td>
<td></td>
</tr>
<tr>
<td>-1.64133</td>
<td>43820</td>
<td></td>
</tr>
<tr>
<td>0.53719</td>
<td>550200</td>
<td></td>
</tr>
<tr>
<td>0.53719</td>
<td>549767</td>
<td></td>
</tr>
<tr>
<td>1.58048</td>
<td>33100</td>
<td></td>
</tr>
<tr>
<td>1.58048</td>
<td>33817</td>
<td></td>
</tr>
<tr>
<td>3.05424</td>
<td>18341</td>
<td></td>
</tr>
<tr>
<td>3.21306</td>
<td>77750</td>
<td></td>
</tr>
<tr>
<td>3.46568</td>
<td>21635</td>
<td>2.00000</td>
</tr>
<tr>
<td>3.79631</td>
<td>79956</td>
<td>2.00000</td>
</tr>
<tr>
<td>4.18602</td>
<td>04350</td>
<td>2.00000</td>
</tr>
<tr>
<td>4.61395</td>
<td>05337</td>
<td>2.00000</td>
</tr>
<tr>
<td>5.05837</td>
<td>84312</td>
<td>2.00000</td>
</tr>
<tr>
<td>5.49755</td>
<td>03169</td>
<td>2.00000</td>
</tr>
<tr>
<td>5.91051</td>
<td>27679</td>
<td>2.00000</td>
</tr>
<tr>
<td>6.27790</td>
<td>84969</td>
<td>2.00000</td>
</tr>
<tr>
<td>6.58272</td>
<td>45938</td>
<td>2.00000</td>
</tr>
<tr>
<td>6.81096</td>
<td>24646</td>
<td>2.00000</td>
</tr>
<tr>
<td>6.95219</td>
<td>74335</td>
<td>2.00000</td>
</tr>
</tbody>
</table>

Mathematisches Institut der Technischen Universität München
Arcisstrasse 21
8000 München 2, Germany