LAPACK 3.1 `xHSEQR`: Tuning and Implementation Notes on the Small Bulge Multi-shift QR Algorithm with Aggressive Early Deflation

Ralph Byers*
405 Snow Hall
Department of Mathematics
University of Kansas
Lawrence, Kansas 66045
USA
byers@math.ku.edu

May 2007

Abstract

This note documents implementation details of the small bulge, multi-shift QR algorithm with aggressive early deflation that appears as LAPACK version 3.1 programs `CHSEQR`, `DHSEQR`, `SHSEQR` and `ZHSEQR` and the subroutines they call. These codes calculate eigenvalues and optionally a Schur factorization of a Hessenberg matrix. They do the bulk of the work required to calculate eigenvalues and optionally eigenvectors of a general non-symmetric matrix. This report is intended to provide some guidance for setting the machine dependent tuning parameters, to help maintainers to identify and correct problems, and to help developers improve upon this implementation.

*Portions of this work were accomplished while this author was on sabbatical leave at and partially supported by the Lawrence Berkeley National Laboratory. This work was also partially supported by by the National Science Foundation under awards 0098150 and 0112375.

1 Introduction

This report explains the structure and some of the implementation details of the small bulge, multi-shift QR algorithm with aggressive, early deflation [4, 5] as it is implemented in LAPACK 3.1 by `CHSEQR`, `DHSEQR`, `SHSEQR` and `ZHSEQR` and the subroutines that they call. These codes calculate eigenvalues and optionally a Schur factorization of a Hessenberg matrix. They do the bulk of the work required to calculate eigenvalues and optionally eigenvectors of a general non-symmetric matrix.

The small bulge multi-shift QR algorithm executes QR sweeps at BLAS 3 [7] speed [4] while aggressive early deflation recognizes and deflates converged eigenvalues earlier in the iteration than does traditional small-subdiagonal deflation [5, 12]. Although we started from a working experimental code and we reused some of the older LAPACK 3.0 code (in `xLAHQR`), nevertheless it took many weeks of full time effort to write, debug and tune a final production version of the new QR algorithm. Improvements to the QR algorithm may well be discovered. It would be a waste to discard this work and rewrite the entire code in order to incorporate an improvement. Likewise, it would be a waste to discard improvements, because it is too difficult and time consuming to modify the existing code. The intention of this report is to explain enough of the structure and implementation details to make it feasible to incorporate improvements by modifying this code.

This report presumes that the reader is already familiar with the small bulge, multi-shift QR algorithm with aggressive, early deflation [4, 5]. For ease of explication we use "the new QR algorithm" to refer to the small bulge, multi-shift QR algorithm with aggressive early deflation [4, 5] as implemented in LAPACK version 3.1. We use "the old QR algorithm" to refer to the large bulge multi-shift QR algorithm [3] as implemented in LAPACK version 3.0. We use "the implicit double shift QR algorithm" to refer to the implicit double shift QR algorithm proposed in the early 1960's [8] modified with the more-robust-against-convergence-failures shift strategy [6] and a more stringent deflation criterion [1]. (The implicit double shift QR algorithm is implemented in LAPACK 3.1 subroutines `DLAHQR` and `SLAHQR`. LAPACK 3.1 subroutines `CLAHQR` and `ZLAHQR` implement an implicit single-shift QR algorithm for complex, Hessenberg matrices.)

The LAPACK 3.1 programs `CHSEQR`, `DHSEQR`, `SHSEQR` and `ZHSEQR` (along with the subroutines they call) implement the new QR algorithm. Specifically, `CHSEQR`, `DHSEQR` and `SHSEQR` (and the subroutines they call) implement the algorithm for the Fortran 77 standard arithmetic types `COMPLEX`, `DOUBLE PRECISION` and `REAL`, respectively. `ZHSEQR` implements the algorithm for the arithmetic type `COMPLEX*16` which is a frequent extension to Fortran 77. Following common convention, this manuscript refers to the four implementations collectively as `xHSEQR`. Similar Lapack subroutine families are also referenced collectively with an `x` in place of a leading `C`, `D`, `S` or `Z`. The old QR algorithm is implemented in Lapack 3.0 as `xHSEQR` . Slightly different versions of the implicit double shift QR algorithm are implemented in Lapack 3.0 and Lapack 3.1 as `DLAHQR` and `SLAHQR`. Both the Lapack 3.0 and Lapack 3.1 versions of `CLAHQR` and `ZLAHQR` use implicit single shifts.

As implemented in Lapack 3.1, the new QR algorithm is substantially more complicated than either the old QR algorithm or the implicit double shift QR algorithm. The new QR algorithm has over 1,300 executable lines of code spread among seven subroutines (not including calls to the BLAS [7, 13, 14], `xGEHRD`, `xLABAD`, `xLACPY`, `xLANV2`, `xLARFG`, `xLARF`, `xLASET`, `DORGHR`, `SORGHR`, `CUNGHR`, `ZUNGHR` and `xTREXC`). The old QR algorithm in Lapack 3.0 has the 335 executable lines spread over two subroutines (not including calls to the BLAS [7, 13, 14], `xLABAD`, `xLACPY`, `xLANV2`, `xLARFG` and `xLARFX`). The implicit double shift QR algorithm, Lapack 3.1 subroutine `xLAHQR` has 191 executable lines of code in one subroutine (not including calls to the BLAS [7, 13, 14], `xLABAD`, `xLARFG` and `xLANV2`).

The implementation of the new QR algorithm, Lapack 3.1 subroutine `xHSEQR` has the following goals:

- To be a "drop in" replacement for the old, Lapack 3.0 version of `xHSEQR`; (In particular both the old, Lapack 3.0 version and the new Lapack 3.1 version have the same calling sequence; operate satisfactorily with the same workspace; and return similar, compatible output. Note that the Lapack 3.0 version and the Lapack 3.1 version may return quite different results due to rounding errors, ill-conditioning and/or eigenvalue ordering along the diagonal of the triangular or quasi-triangular factor.)
- To be robust against overflows and both gradual and sudden underflow;
- To be robust against QR iteration convergence failures; and
- when linked with suitably tuned BLAS, to have efficiency and accuracy comparable to or better than the Lapack 3.0 version of `xHSEQR` for all

matrices and to deliver substantially greater efficiency for matrices of order greater than a few hundred.

The numerical experiments that are briefly mentioned below were conducted on a workstation with a 2GHz Xeon processor, 400MHz front side buss, 512MB Level 2 cache and 2 gigabyte RAM memory running Linux. Numerical experiments concentrated on `DHSEQR`. Code was compiled with g77/gcc version 3.4.6 with options -O3 -malign-double -march=i686 -mcpu=i686 -mtune=pentium4 -mieee-fp -funroll-all-loops -fomit-frame-pointer -fno-trapping-math and linked with GOTO BLAS version 0.9 [10, 11].

In the numerical experiments used for setting the machine dependent tuning parameters, n-by-n matrices were stored in n-by-n arrays. Non-random matrices were selected from the Non-Hermitian Eigenvalue Problems (NEP) Collection [2]. Pseudo-random Hessenberg matrices are matrices which have normally distributed pseudo-random numbers with mean zero and variance one on the diagonal and upper triangle. The subdiagonal entry $(j + 1, j)$ entry has the square root of a Chi-squared distributed random variable with $n - j$ degrees of freedom. Hessenberg matrices with the same distribution of random number entries can also be generated by applying the reduction to Hessenberg form algorithm [9, Algorithm 7.4.2] (modified slightly to give positive subdiagonal entries) to an n-by-n matrix all of whose entries are normally distributed variables with mean zero and variance one.

Of course, there may be a benefit from different choices of the tuning parameters for other computational platforms, other classes of matrices, other floating point precisions, other implementations of the BLAS, other compilers, compiler options or perhaps even other operating systems. Nevertheless, we expect the default tuning parameters to give good performance for a wide variety of matrices in a wide variety of computational environments. This expectation is met, for example, on an Origin2000 computer with 400MHz R12000 processors, 8MB of level 2 cache and optimized BLAS from the SGI/Cray Scientific Library version 1.2.0.0.

2 Outline

The new QR algorithm implementation consists of subroutines `xHSEQR`, `xLAHQR`, `xLAQR0`, `xLAQR1`, `xLAQR2`, `xLAQR3`, `xLAQR4`, `xLAQR5`, and `IPARMQ`. The calling graph in Figure 1 omits calls to the BLAS [7, 13, 14], `IPARMQ`, `xGEHRD`, `xLABAD`, `xLACPY`, `xLANV2`, `xLARFG`, `xLARF`, `xLASET`, `DORGHR`, `SORGHR`, `CUNGHR`, `ZUNGHR` and `xTREXC`.

At least once per multi-shift QR sweep (often more than once) aggressive early deflation [5] calculates a full Schur decomposition of a trailing principal submatrix of a Hessenberg matrix. This trailing principal submatrix may itself be of large enough order to benefit from calling the new QR algorithm algorithm recursively. Unfortunately, the 1977 Fortran standard does not include recursion. Instead of true recursion, `xHSEQR` implements one level of recursion assigning different names to nearly identical subroutines. (Subroutine `xLAQR`4 is a near duplicate of `xLAQR`0 and subroutine `xLAQR`2 is a near duplicate of `xLAQR`3.) The subroutines called at the higher level of recursion have minimum modification to avoid even higher levels of recursion. Numerical experiments with random matrices and matrices from a variety of applications [2] verify that there is little to be gained from higher levels of recursion.

`IPARMQ`: Subroutine `IPARMQ` sets several "tuning" parameters that affect the operation and efficiency of the computation. Tuning parameters are described throughout this manuscript.

`xHSEQR`: Subroutine `xHSEQR` is the "user callable" entrance to the new QR algorithm, although it would typically be called through `xGEEV` or `xGEEVX`. It selects either the implicit double shift QR algorithm or the new QR algorithm depending primarily on the order of the Hessenberg matrix.

`xLAHQR`: Subroutine `xLAHQR` is an implementation of the classic implicit double shift QR algorithm. It is a complete, Hessenberg matrix eigenvalue/Schur factorization subroutine. It is slightly modified version of the subroutine of the same name in LAPACK 3.0. A few lines were modified to be more robust against overflow and underflow. Both the small-compared-to-nearest-diagonal-entries convergence criterion used by the LAPACK 3.0 code and the Ahues and Tisseur [1] convergence criterion must be satisfied in order set a small subdiagonal entry to zero.

`xLAQR0`, `xLAQR4`: Subroutine `xLAQR`0 manages the new, small bulge multishift QR algorithm with aggressive deflation. It is a complete, Hessenberg matrix eigenvalue/Schur factorization subroutine. It adjusts the

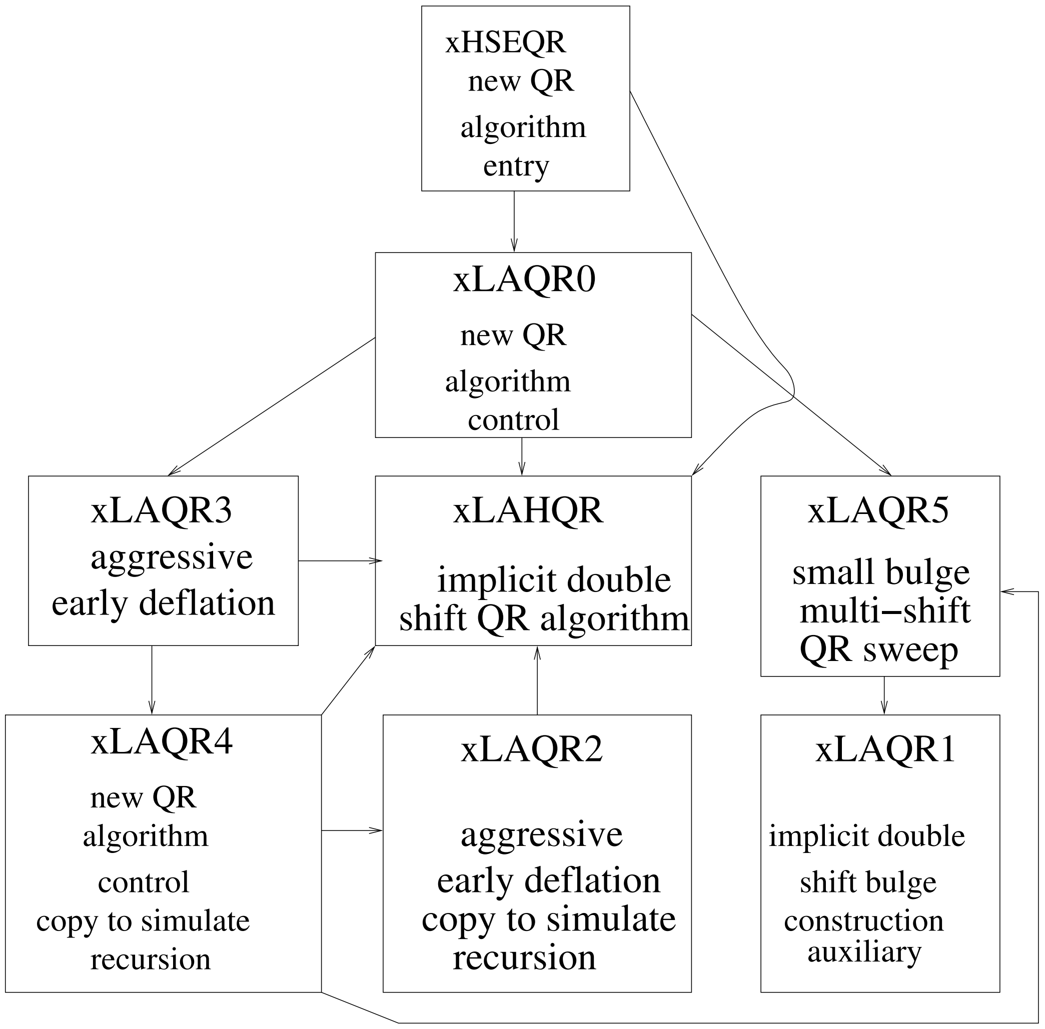

Figure 1: `xHSEQR` calling graph omitting the BLAS, `IPARMQ`, `xGEHRD`, `xLABAD`, `xLACPY`, `xLANV2`, `xLARFG`, `xLARF`, `xLASET`, `DORGHR`, `SORGHR`, `CUNGHR`, `ZUNGHR`, and `xTREXC`. Arrows point from calling subroutine toward a called subroutine. One level of recursion is obtained by nearly duplicating `xLAQR0` as `xLAQR4` and `xLAQR3` as `xLAQR2`.

size of the deflation window, selects the shifts, and determines when to use a multi-shift QR sweep between calls to aggressive early deflation.

Subroutine `xLAQR4` is a near duplicate of `xLAQR0` used to implement one level of recursion. It avoids higher levels of recursion by calling the implicit double shift QR algorithm, `xLAHQR`, instead of itself and by calling `xLAQR2` instead of `xLAQR3` to perform aggressive, early early deflation.

`xLAQR1`: This is an implicit double shift bulge construction auxiliary subroutine. Given a 2-by-2 or 3-by-3 Hessenberg matrix H, and a pair of scalar shifts s_1 and s_2, subroutine `xLAQR1` returns a scalar multiple of the first column of the product $(H - s_1)(H - s_2)$ taking care to avoid overflows and most underflows. This is useful for starting implicit double shift bulges during the QR algorithm.

`xLAQR2`, `xLAQR3`: Subroutine `xLAQR3` performs aggressive, early deflation on a Hessenberg matrix H. It returns the converged eigenvalues it finds (if any) along with the eigenvalues of a trailing principal submatrix. The latter may be used for shifts during a subsequent small bulge, multi-shift QR sweep.

In addition, `xLAQR3` overwrites the Hessenberg matrix H by a similarity transformation of a minute perturbation of H which is still in Hessenberg form. On output, converged eigenvalues (if any are discovered) appear as eigenvalues of isolated 1-by-1 and 2-by-2 blocks along the trailing diagonal entries. (In the complex case there are only isolated 1-by-1 trailing diagonal blocks.) The similarity transformation is optionally accumulated into a given orthogonal (unitary in the complex case) matrix.

Subroutine `xLAQR2` is a near duplicate of `xLAQR3` used to implement one level of recursion. Subroutine `xLAQR2` avoids higher levels of recursion by calling the implicit double shift QR algorithm `xLAHQR` instead of `xLAQR4` or `xLAQR0`.

`xLAQR5`: Subroutine `xLAQR5` performs a small bulge, multi-shift QR sweep. Even in the complex case, it applies the shifts in pairs using implicit double shift bulges. (This simplifies maintaining real and complex versions of the code, and uses less arithmetic work than would be needed for implicit single shift bulges.) Vigilant deflation requires both the small-compared-to-nearest-diagonal-entries criterion used by the LAPACK 3.0 code and the Ahues and Tisseur [1] stopping criterion to be satisfied in order to set a small subdiagonal entry to zero. (The Ahues and Tisseur [1] stopping criterion is applied even though the matrix is perturbed from Hessenberg form by a chain of implicit double shift bulges.)

For ease of reference, Table 1 lists some of the variables discussed in this report and used by `xHSEQR` and/or the subroutines it calls.

3 Details

3.1 `xHSEQR`:

Subroutine `xHSEQR` is the "user callable" entrance to the new QR algorithm, although it would typically be called through `xGEEV` or `xGEEVX`. It sends larger order matrices to `xLAQR0` where they can benefit from the BLAS 3 [7] efficiency and early convergence of the new algorithm. Smaller order matrices do not benefit from the BLAS 3 approach or early convergence. They go directly to `xLAHQR`, the implicit double shift algorithm, to avoid the higher computational overhead of the new algorithm.

The threshold between "larger order matrices" that benefit from the new algorithm and and "smaller order matrices" that are more efficiently factored by the implicit double shift QR algorithm is specified by `NMIN= ILAENV(ISPEC=12)=IPARMQ(ISPEC=12)`, i.e., is specified through `ILAENV` by calling `IPARMQ` with `ISPEC=12`. The minimum valid value of `NMIN` is eleven. There is no maximum valid value. The unmodified, default version of `IPARMQ` returns `IPARMQ`(`ISPEC=12`)=75. This default is obtained from empirical experiments with pseudo-random matrices. A different value of `NMIN` may be appropriate for other classes of matrices or for other computers.

In the rare case of a implicit double shift QR algorithm convergence failure, a small order matrix may be diverted to the new algorithm, because it is more robust against convergence failures. In order to insure that there is sufficient scratch space in the unused (i.e., zero) entries below the first subdiagonal, a Hessenberg matrix smaller than 49-by-49 is copied to a 49-by-49 locally declared scratch array. There is no particular reason to prefer 49-by-49 over similarly sized scratch arrays. The scratch array must be at least 12-by-12 for the new algorithm to apply at all. However, 12-by-12 array allows only

H	The Hessenberg matrix iterate.
IHI ILO	On entry H is assumed to already be triangular in rows and columns 1 through $\mathtt{ILO} - 1$ and $\mathtt{IHI} + 1$ through N
KACC22	Parameter determining the mode of execution in xLAQR5. Set by IPARMQ. See Subsections 3.4 and 3.5.
KBOT KTOP	The active, unreduced Hessenberg submatrix lies in rows and columns KTOP through KBOT.
LD	The number of eigenvalues deflated by a single application of aggressive early deflation.
N	The order of the Hessenberg matrix.
NH	The order of the active, unreduced Hessenberg submatrix, i.e., $\mathtt{KTOP} - \mathtt{KBOT} + 1$
NHo	The number of columns in a horizontal scratch array. See Figures 2 and 3.
NIBBLE	Parameter determining when to skip a multi-shift QR sweep. Set by IPARMQ. See Subsections 3.2 and 3.5.
NMIN	The xLAHQR/xLAQR0 cross over point. Set by IPARMQ. See Subsections 3.2 and 3.5.
NShfts	The number of simultaneous shifts in a multi-shift QR sweep.
NSMAX	The maximum number of simultaneous shifts for which there is sufficient subdiagonal scratch space for xLAQR5.
NSR	The recommended number of simultaneous shifts. Set by IPARMQ. See Subsections 3.2 and 3.5.
NVe	The number of rows in a vertical scratch array. See Figures 2 and 3.
NW	The order of the deflation window.
NWMAX	The maximum order of deflation window for which there is sufficient subdiagonal scratch space for xLAQR2 and/or xLAQR3.
NWR	The recommended deflation window size. Set by IPARMQ. See Subsections 3.2 and 3.5.

Table 1: Cast of characters: some of the variables discussed in this report and used by xHSEQR and/or the subroutines it calls.

two simultaneous shifts and a weak 3-by-3 deflation window, so the new algorithm would not be significantly different than the implicit double shift algorithm. A 49-by-49 array allows six simultaneous shifts and a 16-by-16 deflation window which is empirically more robust against convergence failures than the implicit double shift algorithm. See the discussion of `xLAQR0` below. A larger scratch array would allow more simultaneous shifts and a larger deflation window which might be marginally even more robust against convergence failure.

3.2 xLAQR0 and xLAQR4

Subroutine `xLAQR0` is a complete Hessenberg matrix eigenvalue/Schur factorization algorithm. It controls the size of the deflation window, choice of shifts and determines when a multi-shift QR sweep is performed between applications of aggressive early deflation. Subroutine `xLAQR4` is a near duplicate of `xLAQR0` used to implement one level of recursion. Subroutine `xLAQR4` calls `xLAQR2` instead of `xLAQR3` and calls `xLAHQR` instead of itself.

The new algorithm as implemented in `xLAQR0` uses the "unused" zero entries below the subdiagonal of the Hessenberg iterate for the scratch space required by `xLAQR3` and `xLAQR5`. Figures 2 and 3 show how scratch space is laid out. As implemented here, an n-by-n matrix has enough subdiagonal scratch space to support an $\lfloor (n+6)/9 \rfloor$ (less one, if this quantity is odd) shift small bulge multi-shift QR sweep and an order $\lfloor (n-1)/3 \rfloor$ deflation window. In particular, matrices smaller than 12-by-12 do not have enough subdiagonal scratch space for even the minimal two shifts, so these are sent to `xLAHQR`, the implicit double shift QR algorithm. (However, if `IPARMQ` is unmodified, then it is extraordinarily rare to call `xLAQR0` with a matrix smaller than 75-by-75 and impossible to call `xLAQR0` with a matrix smaller than 49-by-49.)

Input arguments `ILO` and `IHI` indicate that the initial Hessenberg matrix is already in triangular form in rows and columns one through `ILO`-1 and `IHI`+1 through the order of the whole Hessenberg matrix `N`. Hence, the eigenvalue problem or Schur factorization reduces to the Hessenberg matrix in rows and columns `ILO` through `IHI`. Subroutine `xLAQR0` obtains tuning parameters by passing `N`, `ILO`, `IHI` and the dimension of the workspace array `LWORK` to `ILAENV` which passes them along to `IPARMQ` where the tuning parameters are set. These parameters do not change during the algorithm although different tuning parameters may be set in simulated recursive calls.

$$
\begin{array}{cccccccccccccc}
\mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
 & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
 & & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{W} & \mathrm{W} & \mathrm{W} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{W} & \mathrm{W} & \mathrm{W} & & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{W} & \mathrm{W} & \mathrm{W} & & & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{W} & \mathrm{W} & \mathrm{W} & & & & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{W} & \mathrm{W} & \mathrm{W} & & & & & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{W} & \mathrm{W} & \mathrm{W} & & & & & & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{W} & \mathrm{W} & \mathrm{W} & & & & & & & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} & \mathrm{H} \\
\mathrm{V} & \mathrm{V} & \mathrm{V} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{S} & \mathrm{D} & \mathrm{D} & \mathrm{D} \\
\mathrm{V} & \mathrm{V} & \mathrm{V} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & & \mathrm{D} & \mathrm{D} & \mathrm{D} \\
\mathrm{V} & \mathrm{V} & \mathrm{V} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} & & & \mathrm{D} & \mathrm{D}
\end{array}
$$

Figure 2: Arrangement of subdiagonal scratch space for `xLAQR`3 and `xLAQR`2, aggressive early deflation. This is a `N`-by-`N` = 14-by-14 schematic example with an `NW`-by-`NW` = 3-by-3 deflation window. The H's are the nontrivial entries of the Hessenberg QR iterate. The D's indicate the trailing principal submatrix in the deflation window. (As the problem deflates, the position of the deflation window moves up along the subdiagonal. However, the scratch arrays W and T do not move. They remain in the leftmost columns and bottom rows, respectively.) The S is the spike root. The V's form a `NW`-by-`NW` scratch array for the orthogonal (or unitary in the complex case) Schur factor of the principal submatrix in the deflation window. The T's form an `NW`-by-`NHo` horizontal work space for matrix-matrix multiplication where `NHo` $=$ `N` -2`NW` -1 makes the T array as large as possible for this arrangement of the subdiagonal scratch space. The quasi-triangular (or triangular in the complex case) Schur factor is temporarily stored in the left-most `NW` columns of T. The W's form an `NVe`-by-`NW` vertical workspace for matrix-matrix multiplication where `NVe` $=$ `N` $-$ 2`NW` $-$ 1 makes the W array as large as possible for this arrangement of the subdiagonal scratch space. Subroutines `xLAQR`2 and `xLAQR`3 require the width of T to be `NHo` $\geq$ `NW` columns and the height of W to be `NVe` $\geq$ `NW` rows, so there is room for the triangular factor in T and enough scratch space in W and T to achieve BLAS 3 speed matrix-matrix multiplies. With this restriction, there is enough subdiagonal scratch space for an order `NWMAX` $= \lfloor(\mathtt{N}-1)/3\rfloor$ deflation window.

```
H  H  H  H  H  H  H  H  H  H  H  H  H  H
H  H  H  H  H  H  H  H  H  H  H  H  H  H
.  H  H  H  H  H  H  H  H  H  H  H  H  H
.  .  H  H  H  H  H  H  H  H  H  H  H  H
   .  .  H  H  H  H  H  H  H  H  H  H  H
      .  .  H  H  H  H  H  H  H  H  H  H
Y  Y  Y  .  .  H  H  H  H  H  H  H  H  H
Y  Y  Y     .  .  H  H  H  H  H  H  H  H
Y  Y  Y        .  .  H  H  H  H  H  H  H
Y  Y  Y           .  .  H  H  H  H  H  H
Y  Y  Y              .  .  H  H  H  H  H
U  U  U  W  W  W  W  W  .  .  H  H  H  H
U  U  U  W  W  W  W  W     .  .  H  H  H
U  U  U  W  W  W  W  W        .  .  H  H
```

Figure 3: Arrangement of subdiagonal scratch space for `xLAQR5`, the small bulge multi-shift QR sweep. This is an `N`-by-`N` = 14-by-14 schematic example with only `NShfts` = 2 simultaneous shifts. The H's represent the nontrivial elements of the Hessenberg QR iterate. The two subdiagonal boarder indicated with dots is used for bulge chasing. The U's form a (3`NShfts` − 3)-by-(3`NShfts` − 3) work array for the accumulated product of elementary reflectors. The W's form a (3`NShfts` − 3)-by-`NHo` work array for matrix-matrix multiplication where `NHo` = `N` − 6`NShfts` + 3 makes the W array as large as possible for this arrangement of the subdiagonal scratch space. The Y's form a `NVe`-by-(3`NShfts` − 3) work array also for matrix-matrix multiplication where `NVe` = `N` − 6`NShfts` + 3 makes the Y array as large as possible for this arrangement of the subdiagonal scratch space. (As the problem deflates, some of the two subdiagonal boarder becomes unused. The W array and/or the Y array could be up to two columns or rows larger, respectively. However, this is unimplemented.) Subroutine `xLAQR5` arbitrarily requires there to be `NVe` $\geq$ (3`NShfts` − 3) rows in Y and `NHo` $\geq$ (3`NShfts` − 3) columns in W so that there is enough scratch space to allow BLAS 3 speed matrix-matrix multiplies. With this restriction, `NShfts` may be as large as $\lfloor(\mathtt{N}+6)/9\rfloor$ (less one, if this quantity is odd).

Subroutine `xLAQR0` obtains the following tuning parameters from `ILAENV`/`IPARMQ`.

`NSR`: `NSR=ILAENV(ISPEC=15)=IPARMQ(ISPEC=15)`, the recommended number of simultaneous shifts. To be valid, `NSR` must be positive and even (even in the complex case). The largest possible number of shifts for which there is sufficient subdiagonal workspace is $\mathtt{NSMAX} = \lfloor(\mathtt{N}+6)/9\rfloor$ (less one, if this quantity is odd) where `N` is the order of the Hessenberg matrix.

If `IPARMQ` is unmodified, then `IPARMQ(ISPEC=15)` returns a value as specified in Table 2. These choices for the recommended number of shifts were developed from numerical experiments with pseudo-random Hessenberg matrices. Consequently, it is not unlikely that other computers or other classes of matrices would benefit from different choices of the recommended number of shifts.

Note that except in case of a rare QR convergence the double implicit shift algorithm (single implicit shift in the complex case) `xLAHQR` which does not reference `NSR` is used for matrices of order less than or equal to `NMIN`. The parameter `NMIN` is set by the unmodified `IPARMQ` to $\mathtt{NMIN} = 75$.

It is not unusual that the actual number of simultaneous shifts in a small bulge multi-shift QR sweep may be less than `NSR`. See below.

`NMIN`: `NMIN=ILAENV(ISPEC=12)=IPARMQ(ISPEC=12)`, the threshold matrix order at or below which `xLAHQR`, the implicit double shift algorithm (implicit single shift algorithm in the complex case), is preferred to `xLAQR0`, the new algorithm. To be valid, `NMIN` must be greater than or equal to eleven.

If `IPARMQ` is unmodified, then `ILAENV(ISPEC=12)=IPARMQ(ISPEC=12)` returns the default value of 75.

`NIBBLE`: `NIBBLE=ILAENV(ISPEC=14)=IPARMQ(ISPEC=14)`, a threshold controlling whether or not to call `xLAQR5`, the small bulge multi-shift QR sweep, between two successive calls to `xLAQR3`, aggressive early deflation. After a call to `xLAQR3`, `xLAQR0` skips a call to `xLAQR5`, if there is a heuristic reason to expect that a subsequent call to `xLAQR3` will find many converged eigenvalues without an intervening call to `xLAQR5`. A

If $\texttt{IHI} - \texttt{ILO} + 1$ is greater than or equal to ...	... but less than ...	... IPARMQ(ISPEC=15) returns ...
0	30	2
30	60	4
60	150	10
150	590	$\left\lfloor \frac{\texttt{IHI}-\texttt{ILO}+1}{\log_2(\texttt{IHI}-\texttt{ILO}+1)} \right\rfloor$
590	3000	64
3000	6000	128
6000	∞	256

Table 2: If IPARMQ is unmodified, then IPARMQ(ISPEC=15), the default recommended number of simultaneous shifts, is specified in this table. If the *ad-hoc*, increasing function used for $150 \leq \texttt{IHI} - \texttt{ILO} + 1 < 590$ is odd, then its value is decreased by one. Note that except in case of a rare QR convergence failure, the double implicit shift algorithm (single implicit shift in the complex case) xLAHQR which does not reference NSR is used for matrices of order less than or equal to NMIN. The parameter NMIN is set by the unmodified IPARMQ to $\texttt{NMIN} = 75$.

QR sweep is skipped if the last call to `xLAQR3` discovered at least one converged eigenvalue and either of the following occur.

- Skip the next call to `xLAQR5`, if the active unreduced Hessenberg submatrix fits entirely within the next deflation window. In this case, if there is no QR iteration convergence failure, then the next call to `xLAQR3` will discover all the eigenvalues of the active block without an intervening call to `xLAQR5`. (Convergence failures are rare.)
- Skip the next call to `xLAQR5` if the last call to `xLAQR3` used deflation window size `NW` and found more than $\left(\frac{\texttt{NIBBLE}}{100}\right)\times\texttt{NW}$ converged eigenvalues. In particular, if $\texttt{NIBBLE} \leq 0$, then `xLAQR0` skips `xLAQR5` except if the last call to `xLAQR3` found no converged eigenvalues. If $\texttt{NIBBLE} \geq 100$, then `xLAQR0` always calls `xLAQR5` between successive calls to `xLAQR3` except in case the active Hessenberg principal submatrix fits entirely within the next deflation window.

If `IPARMQ` is unmodified, then `ILAENV(ISPEC=14)=IPARMQ(ISPEC=14)` returns the default value of $\texttt{NIBBLE} = 14$. (This is not a typographical error: both `ISPEC` and the default return value are 14.) This small value of `NIBBLE` reflects the expectation that the computational cost of performing a small bulge multi-shift QR sweep with `xLAQR5` is substantially greater than the cost of looking through the deflation window with `xLAQR3`. In some computational environments, e.g., one with well-tuned parallel BLAS, a larger value of `NIBBLE` may be appropriate.

`NWR`: `NWR=ILAENV(ISPEC=13)=IPARMQ(ISPEC=13)`, the recommended deflation window size. To be valid, `NWR` must be positive, but for heuristic reasons, the code requires that the deflation window size be greater than or equal to two. The largest possible deflation window size for which there is sufficient subdiagonal scratch space is $\texttt{NWMAX} = \lfloor(\texttt{N} - 1)/3\rfloor$ where `N` is the order of the Hessenberg matrix. The actual deflation window size may differ from `NWR` as described below.

If `IPARMQ` is unmodified, then `ILAENV(ISPEC=13)=IPARMQ(ISPEC=13)` returns the default value of $\texttt{NWR} = \texttt{NSR}$ for matrices of order less than or equal to 500 and $\texttt{NWR} = 3\texttt{NSR}/2$ for matrices of larger order.

`KACC22`: `KACC22=ILAENV(ISPEC=16)=IPARMQ(ISPEC=16)`, recommended computational path to follow in `xLAQR5`. Subroutine `xLAQR0` simply passes this parameter into xLAQR5 (which keeps a call to ILAENV and IPARMQ out of the main loop). KACC22 is described below in Subsection 3.4.

At the beginning of each iteration, xLAQR0 examines the current Hessenberg iterate and locates an unreduced Hessenberg principal submatrix setting KTOP and KBOT to the first and last row index, respectively. Hence, the active block has order $\mathtt{NH} = \mathtt{KTOP} - \mathtt{KBOT} + 1$ and occupies rows and columns KTOP through KBOT.

Ordinarily, the deflation window size NW is set as follows.

- If $\mathtt{NH} \leq \min(\mathtt{NMIN}, \mathtt{NWMAX})$, then $\mathtt{NW} = \mathtt{NH}$. With this choice, the entire active unreduced Hessenberg block fits in the deflation window. If there is no QR convergence failure, then the next call to xLAQR3, aggressive early deflation, will use xLAHQR to discover all the eigenvalues in the active block. This choice reflects the heuristic expectation that matrices of order NMIN or less are more efficiently processed by xLAHQR, the implicit double shift algorithm than by the new algorithm.

- Let H be the current Hessenberg iterate and let $\mathtt{L} = \min(\mathtt{NWR}, \mathtt{NH}, \mathtt{NWMAX})$. Ordinarily, if $\mathtt{NH} > \min(\mathtt{NMIN}, \mathtt{NWMAX})$, then the deflation window size is set to L or $\mathtt{L}+1$ depending upon which choice corresponds to the smaller of $|\mathtt{H}(\mathtt{KBOT} - \mathtt{L} + 1, \mathtt{KBOT} - \mathtt{L})|$ and $|\mathtt{H}(\mathtt{KBOT} - \mathtt{L}, \mathtt{KBOT} - \mathtt{L} - 1)|$.

  This choice is motivated by the following heuristic. The aggressive deflation spike "hangs" from and is scaled by the $(\mathtt{KBOT}-\mathtt{NW}+1, \mathtt{KBOT}-\mathtt{NW})$ subdiagonal entry. The above choice selects the subdiagonal entry of smaller magnitude from two neighbors. Heuristically, the smaller subdiagonal entry helps discover converged eigenvalues by scaling the spike by a smaller number. Searching through just two neighboring subdiagonal entries reflects the fact that complex eigenvalues, i.e., eigenvalues with nonzero imaginary part, cause the real quasi-triangular Schur factor of a real matrix to have nontrivial 2-by-2 bumps. A poor choice of NW might hang the spike from a relatively large magnitude subdiagonal entry that is part of a converging 2-by-2 bulge. The heuristic assumes that a much smaller magnitude neighboring subdiagonal would be a better choice.

  There might be a benefit from searching for the smallest magnitude of several neighboring subdiagonal entries. Experiments with pseudorandom Hessenberg matrices indicate that there is little to be gained from this, but it is possible that there is a greater benefit for other classes of matrices.

Experiments with many classes of matrices show that it is unusual for a call to xLAQR3, aggressive early deflation, to fail to discover some converged eigenvalues. However, in rare cases convergence may be slow or stagnant and calls to xLAQR3 do not discover converged eigenvalues. Subroutine xLAQR0 adopts the following strategies to try to break away from stagnant convergence. If xLAQR3 discovers no converged eigenvalues in five successive calls, then, because larger deflation windows tend to be more effective than smaller ones, xLAQR0 doubles the size of the deflation window. If xLAQR3 still finds no converged eigenvalue, then xLAQR0 continues to double the deflation size until it reaches $\min(\mathtt{NH}, \mathtt{NWMAX})$. If xLAQR3 still finds no converged eigenvalues, it reduces the deflation window size by one to try to break up the symmetries that lead to convergence failures.

In addition, if xLAQR3 discovers no converged eigenvalues in a multiple of six successive calls, a set of arbitrary, *ad hoc* shifts are used in place of shifts chosen from a trailing principal submatrix.

The unconverged eigenvalues returned by xLAQR3 are eigenvalues of a trailing principal submatrix of the active block. Empirically, these make good shifts for a small bulge multi-shift QR sweep. Depending on the deflation window size, there may be more shifts, as many shifts or fewer shifts than the recommended NSR. If there are more shifts, xLAQR0 arbitrarily uses the smallest NSR of them to minimize the variation of the shifts from one iteration to the next. (Five different shift strategies are discussed by Bai and Demmel [3] in the context of the large bulge multi-shift QR Algorithm. The shift strategy used by xLAQR0 is closest to Bai and Demmel's S5. An awkward, untested alternative that is even closer to Bai and Demmel's S5 would be to use the NSR shifts that are closest to the previous set of shifts.)

If there are between NSR/2 and NSR shifts, then xLAQR0 uses all of them (possibly discarding one if this number is odd) which results in a small bulge multi-shift QR sweep with fewer than the recommended NSR shifts.

If IPARMQ is unmodified (and there is no QR iteration convergence failure calculating the Schur decomposition of the deflation window in xLAQR3), then there will be at least $(1 - \mathtt{NIBBLE}/100) \times \mathtt{NW} \geq 0.86 \times \mathtt{NSR}$ shifts, i.e., at worst there will be 86% of the recommended NSR shifts.

If IPARMQ has been modified or there is a rare QR iteration convergence failure in xLAQR3, then it is possible that there are fewer than NSR/2 shifts from xLAQR3. In this case, xLAQR0 uses the eigenvalues of an NSR-by-NSR trailing principal submatrix as calculated by xLAQR4 or xLAHQR depending on NMIN and the number of shifts requested. In the rare case of a QR convergence failure while calculating the shifts, xLAQR0 uses the eigenvalues that did converge as shifts or, if none converged, it uses the eigenvalues of the trailing 2-by-2 principal submatrix.

An unimplemented alternative strategy in case of a convergence failure calculating shifts would be to use diagonal entries of the unconverged, would-be-triangular factor as was done in the LAPACK 3.0 version of xHSEQR. However, convergence failures are very rare and it takes two convergence failures to bring execution through this section of code: one calculating the Schur decomposition of the deflation window in xLAQR3 and one calculating shifts in xLAQR0.

3.3 xLAQR2 and xLAQR3:

Subroutine xLAQR3 implements aggressive early deflation [5]. It performs a similarity transformation of a minute perturbation of the Hessenberg iterate to obtain a Hessenberg matrix which (it is to be hoped) has more zeros among the subdiagonal entries and displays some converged eigenvalues in 1-by-1 and 2-by-2 blocks along the trailing diagonal entries. (In the complex case, converged eigenvalues appear in 1-by-1 blocks only.) Subroutine xLAQR2 is a near duplicate of xLAQR3 used to implement one level of recursion.

Aggressive early deflation is illustrated by Figure 4.

In the notation of Figure 4, the spirit of the xLAQR3 convergence criterion is to set a trailing spike element to zero if it is rounding error small compared to the magnitude of the trailing 1-by-1 or 2-by-2 diagonal block of T. (If the corresponding diagonal block of T is zero and the trailing spike element is rounding error small compared to the norm of the spike itself, then the tailing spike element is set to zero.) This mimics the small-compared-to-nearby-elements convergence criterion of the LAPACK 3.0 version of xHSEQR. In particular, it often allows xHSEQR to calculate eigenvalues of graded matrices to high relative accuracy.

To be precise, subroutine xLAQR3 uses the following criteria to determine whether to set a tailing spike entry to zero. Let μ be the machine precision returned by xLAMCH('P') and let σ be the near underflow quantity $\sigma =$ xLAMCH('S')N$/\mu$ where N is the order of the Hessenberg matrix. (The value of σ may be modified through a call to DLABAD or SLABAD on some

$$\begin{bmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & V \end{bmatrix}^T \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ 0 & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & V \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13}V \\ H_{21} & H_{22} & H_{23}V \\ 0 & s & T \end{bmatrix}$$

$$= \left[\begin{array}{ccc|c|ccccc}
\ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\ddots & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} \\
 & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} \\
\hline
 & & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} \\
\hline
 & & & s_1 & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} \\
 & & & s_2 & & \mathtt{x} & \mathtt{x} & \mathtt{x} & \mathtt{x} \\
 & & & \epsilon & & & \mathtt{x} & \mathtt{x} & \mathtt{x} \\
 & & & \epsilon & & & \mathtt{x} & \mathtt{x} & \mathtt{x} \\
 & & & 0 & & & & & \mathtt{x}
\end{array}\right].$$

Figure 4: Illustration of aggressive early deflation. The deflation window is the trailing `NW`-by-`NW` principal submatrix H_{33}. Its Schur factorization $H_{33}V = VT$ where, in the real case, V is orthogonal and T is quasi-triangular and, in the complex case, V is unitary and T is triangular. The orthogonal similarity transformation illustrated in this figure puts the quasi-triangular (triangular in the complex case) factor T into the trailing `NW`-by-`NW` principal submatrix and fills in a spike s immediately to its left. Often some trailing entries of the spike are small enough to be safely set to zero revealing converged eigenvalues in the corresponding trailing 1-by-1 and 2-by-2 blocks of T. If the trailing entries of the spike s are not small enough to be set to zero, then a further similarity transformation that reorders the eigenvalues along the diagonal of T and correspondingly modifies the spike s sometimes gives trailing spike entries that can be set to zero.

machines.) Let the tailing spike element be s_k.

- If the trailing block of the quasi-triangular matrix T is a 1-by-1 block representing an eigenvalue $\lambda \neq 0$ of T, and if the tailing spike element s_k satisfies
$$|s_k| \leq \max(\sigma, \mu\,|\lambda|),$$
then s_k is set to zero and λ is accepted as a converged eigenvalue.

- If the trailing block of the quasi-triangular matrix T is a 1-by-1 block representing an eigenvalue $\lambda = 0$ of T, and if the tailing spike element s_k satisfies
$$|s_k| \leq \max(\sigma, \mu\,\|s\|_2),$$
then s_k is set to zero and $\lambda = 0$ is accepted as a converged eigenvalue.

- If the trailing block of the quasi-triangular matrix T is a 2-by-2 block representing a complex conjugate pair of eigenvalues $\alpha+\beta i$, $\alpha-\beta i$ with α, $\beta \in \mathbf{R}$, and if the trailing two spike elements s_k and s_{k-1} satisfy
$$\max(|s_k|, |s_{k-1}|) \leq \max(\sigma, \mu(|\alpha| + |\beta|)),$$
then both s_k and s_{k-1} are set to zero and both $\alpha+\beta i$ and $\alpha-\beta i$ are accepted as converged eigenvalues.

In the complex case, the absolute values in the first two convergence criteria are obtained by interpreting $|a+bi|$ for a, $b \in \mathbf{R}$ as $|a|+|b|$. In the complex case, there are no 2-by-2 diagonal blocks along the diagonal of T, so the third convergence criterion does not occur.

Subroutine `xLAQR3` reorders the eigenvalues in T using an orthogonal (unitary in the complex case) similarity transformation that also modifies the spike. Each eigenvalue of T takes a turn in the trailing 1-by-1 or 2-by-2 block with an opportunity to be accepted as a converged eigenvalue if the corresponding trailing entries of the spike are small enough.

A final orthogonal (or unitary) similarity transformation is used to order the converged eigenvalues in decreasing order by magnitude along the diagonal. This helps preserve row or column grading if it is present and helps maintain high accuracy in graded matrices.

Ordinarily, if `xLAQR3` finds no converged eigenvalues, then it does not modify the deflation window. This avoids a little rounding error that would result from extracting the Schur decomposition of the deflation window and then returning the window to Hessenberg form. However, there is a rare exception. If the spike is zero but the Schur decomposition of the deflation window is unavailable due to a QR convergence failure, then the deflation window is replaced by the final unconverged Hessenberg QR iterate (and the Hessenberg matrix and converging Schur vectors if required are updated appropriately). This helps promote ultimate convergence in those cases that the convergence failure is due to a case of slow convergence.

3.4 xLAQR5:

Subroutine xLAQR5 implements a small bulge multi-shift QR sweep [4]. Using the set of NShfts shifts selected and passed in by xLAQR0 or xLAQR4, xLAQR5 forms a chain of NShfts/2 implicit double shift bulges that it chases 3NShfts/2 − 2 columns at a time using 3-by-3 elementary reflectors. (For efficiency of execution and ease of maintenance implicit double shift bulges are used even in the complex case.) Depending on the parameter KACC22 described below, it may accumulate each set of reflectors into an orthogonal matrix for efficient BLAS-3 updating of the far-from-diagonal elements of the Hessenberg matrix and updating the converging Schur vectors if these are required.

If NShfts is odd, then one shift is discarded. (In fact, NShfts is never odd. This is a minor example of defensive programming.) In the case of DLAQR5 and SLAQR5, it is a real shift that is discarded. If the resulting number of shifts is not positive, then nothing is done. There is no restriction on how large NShfts may be; it may even be larger than the order of the active block of the Hessenberg matrix.

As it chases chains of 3-by-3 implicit double shift bulges along the diagonal of the Hessenberg matrix, xLAQR5 checks for vigilant deflation, i.e., it tests for small subdiagonals between the bulges that can be safely set to zero causing the problem to deflate. This test is performed just before forming the last row of an implicit double shift bulge. Both the small-compared-to-nearest-diagonal-entries criterion used by the LAPACK 3.0 code and the Ahues and Tisseur [1] stopping criterion must be satisfied in order to set a small subdiagonal entry to zero.

An implicit double shift bulge may collapse into a single shift bulge or a zero shift (non)bulge either by encountering a zero subdiagonal entry (due to vigilant deflation) or by destructive underflow [15, 16]. In either case it is important to get the bulge started again or the corresponding shifts will be lost before they reach the lower-right-hand-corner where intuition and experience suggest they will have their greatest effect. As explained in [4], a bulge that collapses into a zero subdiagonal entry is restarted in the same way that bulges are initially formed, i.e., by carefully calculating a 3-by-3 elementary reflector from a convenient scalar multiple of the corresponding three entries of $(H - s_1 I)(H - s_2 I)$ where s_1 and s_2 are the shifts.

A bulge that collapses due to destructive underflow can sometimes also be restarted using a method similar to the two-small-subdiagonal modification of the QR algorithm described in [17, Pages 535–537]. Figure 5 illustrates. The four subfigures in Figure 5 illustrate a implicit double shift bulge in a Hessenberg matrix H as it collapses due to destructive underflow and as `xLAQR5` attempts to restore it through four successive bulge chasing steps. Subroutine `xLAQR5` calculates a principal elementary reflector and an alternative elementary reflector for the matrix in Subfigure 5(c). The principal one is chosen to reflect the vector $[y_1, y_2, 0]^T$ to a scalar multiple of the first column of I. It is the reflector that would ordinarily be used to chase the collapsed bulge one row down and left one column to the left. The alternative reflects the first column of $(Z - s_1 I)(Z - s_2 I)$ onto a scalar multiple of the first column of I. Here s_1 and s_2 are the shifts and Z is the Hessenberg submatrix indicated by the z's in Subfigure 5(c). If y_1 and y_2 were zero, then the alternate reflector would be the natural choice to restore a new implicit double shift bulge with the given shifts, because it continues the implicit QR factorization of $(H - s_1 I)(H - s_2 I)$. If y_1 or y_2 are not zero, then using the alternative reflection causes two fill-ins indicated by the f's in Subfigure 5(d). If the f's are small enough to be safely set to zero, then the collapsed bulge is restored (unless it too suffers destructive underflow). If the f's are not small enough to be safely set to zero, then the principal reflector is used instead. In this case, the bulge remains collapsed, but `xLAQR5` attempts to restore it again from its new position.

Subroutine `xLAQR5` operates in one of three modes depending on the parameter `KACC22`.

`KACC22` $= 0$: `xLAQR5` does not accumulate reflections and does not use matrix-matrix multiply to update the far-from-diagonal matrix entries. The entire small bulge multi-shift QR sweep is performed using 3-by-3 elementary reflectors. This mode uses the smallest number of floating point operations.

`KACC22` $= 1$: `xLAQR5` accumulates reflections and uses matrix-matrix multiply to up-

$$\begin{array}{ccccccccc}
x & x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x & x \\
 & & & & x & x & x & x & x \\
 & & & & & x & x & x & x \\
 & & & & & & x & x & x \\
 & & & & & & & x & x
\end{array}$$

(a) Uncollapsed implicit double shift bulge.

$$\begin{array}{ccccccccc}
x & x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x & x \\
 & & x & x & x & x & x & x & x \\
 & & x & x & x & x & x & x & x \\
 & & 0 & x & x & x & x & x & x \\
 & & & & & x & x & x & x \\
 & & & & & & x & x & x \\
 & & & & & & & x & x
\end{array}$$

(b) Implicit double shift bulge begins to collapse. The zero comes from setting a destructive underflow quietly to zero.

$$\begin{array}{ccccccccc}
x & x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x & x \\
 & & x & x & x & x & x & x & x \\
 & & & y_1 & z & z & z & x & x \\
 & & & y_2 & z & z & z & x & x \\
 & & & 0 & 0 & z & z & x & x \\
 & & & & & & x & x & x \\
 & & & & & & & x & x
\end{array}$$

(c) Collapsed Implicit double shift bulge. The zeros are a consequence of the underflow in Subfigure 5(b). An alternative reflector is chosen to map the first column of $(Z - s_1 I)(Z - s_2 I)$ onto a scalar multiple of the first column of I. Here s_1 and s_2 are the shifts and Z is the Hessenberg submatrix indicated by the z's in this subfigure.

$$\begin{array}{ccccccccc}
x & x & x & x & x & x & x & x & x \\
x & x & x & x & x & x & x & x & x \\
 & x & x & x & x & x & x & x & x \\
 & & x & x & x & x & x & x & x \\
 & & & x & x & x & x & x & x \\
 & & & f & x & x & x & x & x \\
 & & & f & x & x & x & x & x \\
 & & & & x & x & x & x & x \\
 & & & & & & & x & x
\end{array}$$

(d) Attempt to restore a collapsed implicit double shift bulge. Two fill-ins indicated by f's result from using the alternative reflection. If the fill-ins are small enough to be safely set to zero, then the bulge is restored (unless it also suffers destructive underflow). If not, then the alternative reflector is abandoned in favor of the principal reflector.

Figure 5: The four subfigures show a implicit double shift bulge as it collapses and `xLAQR5` attempts to restore it through four successive bulge chasing steps.

date the far-from-diagonal matrix entries, but it does not take advantage of the block 2-by-2 structure of the accumulated orthogonal factor (unitary factor in the complex case). This mode uses the greatest number of floating point operations, but it has the least complicated implementation and the largest order matrix-matrix multiplications.

KACC22 = 2: xLAQR5 accumulates reflections and takes advantage of 2-by-2 block structure during matrix-matrix multiplies to update the far-from-diagonal matrix entries.

The parameter KACC22 is passed in from xLAQR0 or xLAQR4. It is originally set by KACC22=ILAENV(ISPEC=16)=IPARM(ISPEC=16). The unmodified version of IPARMQ returns the following defaults in terms of the default recommended number of shifts NSR.

If NSR < 14,	then IPARMQ(ISPEC=16) returns 0
If NSR ≥ 14,	then IPARMQ(ISPEC=16) returns 2

If the BLAS-3 subroutines are not well tuned to the underlying computational environment (like the untuned Fortran model implementation), then KACC22 $= 0$ may be the most efficient choice, because it minimizes the amount of arithmetic work and there may be little gained from calling untuned BLAS-3 subroutines. Similarly, if NShfts is small, then the matrix multiplies in xLAQR5 are too small to benefit from a BLAS-3 implementation and KACC22 $= 0$ may be more efficient. If the BLAS-3 subroutines are well tuned to the computational environment but the BLAS-3 subroutine xTRMM is substantially slower than xGEMM, then, despite the greater arithmetic work, avoiding calls to xTRMM by setting KACC22 $= 1$ may be more efficient than KACC22 $= 2$.

3.5 IPARMQ:

Function subroutine IPARMQ is called by ILAENV to set the tuning parameters. Mimicking ILAENV, IPARMQ takes the following input arguments.

ISPEC: Integer specifying which tuning parameter to return.

NAME: Character string specifying which subroutine is requesting a tuning parameter.

OPTS: Character string specifying the options to the calling subroutine. This is a concatenation of the first character of the option strings passed to the calling program.

N: Order of the Hessenberg matrix whose eigenvalues and (optionally) whose Schur factorization are required.

ILO, IHI: The initial active block. It is assumed that the Hessenberg matrix is upper triangular in rows and columns one through ILO − 1 and IHI + 1 through N.

LWORK: The amount of scratch space passed into the calling subroutine.

The original, unmodified version of IPARMQ uses only ISPEC, ILO and IHI.

Better choices of the tuning parameters may depend on the values of the other parameters. Even better tuning parameters might be determined by inspecting the entire Hessenberg matrix, but this is not implemented.

As implemented in Lapack 3.1, tuning parameters do not change during the computation. However the Hessenberg matrix undergoes QR sweeps and deflation during the computation, so there may be some advantage to varying the tuning parameters during the computation. A modification in this direction calls for moving calls to ILAENV into the main loop.

The tuning parameters are discussed in detail above in the context in which they are used. The summary below collects all the tuning parameters in one place.

NMIN=ILAENV(ISPEC=12)=IPARM(ISPEC=12): NMIN is the crossover point between xLAHQR and xLAQR0/xLAQR4. Matrices of order NMIN or less are sent directly to xLAHQR, the implicit double shift QR algorithm. Larger order matrices (and xLAHQR failures) go to xLAQR0 or xLAQR4. NMIN must be at least 11. The default is NMIN = 75.

NWR=ILAENV(ISPEC=13)=IPARM(ISPEC=13): NWR is the recommended deflation window size. NWR is best set greater than or equal to the recommended number of simultaneous shifts NSR, so xLAQR3 and/or xLAQR2 produce a sufficient number of shifts. (See ISPEC=15 below.) If IHI − ILO + 1 $\leq$ 500, then the default value is NWR = NSR. Otherwise, the default value is NWR = 3NSR/2. Larger deflation windows are more powerful and tend to find more converged eigenvalues, but they require computational work. Larger matrices benefit from larger deflation windows.

`NIBBLE=ILAENV(ISPEC=14)=IPARM(ISPEC=14):` Let `LD` be the number of converged eigenvalues discovered by the last call to `xLAQR`3 or `xLAQR`2 and let `NW` be the order of the deflation window. If `LD` is positive and `LD` $>$ (`NW` $\times$ `NIBBLE`)/100, then the next small bulge multi-shift QR sweep is skipped and early deflation is applied immediately to the remaining active diagonal block. Setting `NIBBLE` $= 0$ causes `xLAQR`0 and `xLAQR`4 to skip a multi-shift QR sweep whenever early deflation finds a converged eigenvalue. Setting `NIBBLE` greater than or equal to 100 prevents almost all skipping multi-shift QR sweep. (See Section 3.2 for a complete description of `NIBBLE`.) The default is `NIBBLE` $= 14$. A particularly efficient implementation of the small bulge multi-shift QR sweep `xLAQR`5 (perhaps using well tuned parallel BLAS subroutines) and/or especially large deflation windows may call for a larger value of `NIBBLE`.

`NSR=ILAENV(ISPEC=15)=IPARM(ISPEC=15):` `NSR` is the recommended number of simultaneous shifts to use in a multi-shift QR sweep. This parameter affects performance in a complicated and ill-understood way. Larger values of `NSR` lead to larger matrix-matrix multiplications which can be executed efficiently by well-tuned BLAS. However, larger values of `NSR` also reduces the amount of the computation that does run as matrix-matrix multiplications. In addition, the choice of `NSR` typically affects the other tuning parameters, e.g., typically `NWR` $\geq$ `NSR`. The choice of `NSR` also affects convergence patterns.

See Table 2 for the defaults.

`KACC22=ILAENV(ISPEC=16)=IPARM(ISPEC=16):` `KACC22` determines the mode of execution in `xLAQR`5.

`KACC22=0`: During the multi-shift QR sweep, `xLAQR`5 does not accumulate reflections and does not use matrix-matrix multiply to update the far-from-diagonal matrix entries. This may be a good choice in the absence of well-tuned BLAS subroutines or when the number of simultaneous shifts is small. This choice performs the fewest floating point operations.

`KACC22=1`: During the multi-shift QR sweep, `xLAQR`5 accumulates reflections and uses matrix-matrix multiply to update the far-from-diagonal matrix entries. This may be a good choice if the BLAS-3 triangular-by-full matrix multiplication subroutine `xTRMM` is significantly slower than the full-by-full matrix multiplication subroutine `xGEMM`. This choice does the most floating point operations.

`KACC22=2`: During the multi-shift QR sweep. `xLAQR5` accumulates reflections and takes advantage of 2-by-2 block structure during matrix-matrix multiplies. This may be a good choice if the BLAS-3 triangular-by-full matrix multiplication subroutine `xTRMM` is faster than the full-by-full matrix multiplication subroutine `xGEMM`.

The unmodified version of `IPARMQ` returns the following defaults in terms of the default recommended number of shifts `NSR`.

If `NSR` < 14,	then `IPARMQ(ISPEC=16)` returns 0
If `NSR` ≥ 14,	then `IPARMQ(ISPEC=16)` returns 2

4 Conclusion.

This dreary report explains the structure and some of the implementation details of the small bulge, multi-shift QR algorithm with aggressive, early deflation [4, 5] as it is implemented in LAPACK 3.1 in `CHSEQR`, `DHSEQR`, `SHSEQR` and `ZHSEQR` and the subroutines that they call. These codes calculate eigenvalues and optionally a Schur factorization of a Hessenberg matrix. They do the majority of the work required to calculate eigenvalues and optionally eigenvectors of a general non-symmetric matrix.

This report provides some guidance on how to adjust the tuning parameters provided by `IPARMQ` for maximum performance. The code itself is well commented. It is to be hoped that enough of the overall design and structure of the codes is presented here to allow improvements to be incorporated without having to rewrite large parts of the code.

Acknowledgments:

The author is grateful to Daniel Kressner and Julien Langou for helpful comments on an early draft of this report.

References

[1] Mario Ahues and Françoise Tisseur. A new deflation criterion for the QR algorithm. Technical Report CS-97-353, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, March 1997. LAPACK Working Note 122.

[2] Z. Bai, D. Day, J. Demmel, and J. Dongarra. A test matrix collection for non-Hermitian eigenvalue problems. Prof. Z. Bai, Dept. of Mathematics, 751 Patterson Office Tower, University of Kentucky, Lexington, KY 40506-0027. Also available online from `http://math.nist.gov/MatrixMarket`.

[3] Z. Bai and J. Demmel. On a block implementation of Hessenberg QR iteration. *Intl. J. of High Speed Comput.*, 1:97–112, 1989. Also available online as LAPACK Working Note 8 from `http://www.netlib.org/lapack/lawns/lawn08.ps` and `http://www.netlib.org/lapack/lawnspdf/lawn08.pdf`.

[4] Karen Braman, Ralph Byers, and Roy Mathias. The multi-shift QR algorithm Part I: Maintaining well focused shifts, and level 3 performance. *SIAM J. Matrix Anal. Appl.*, 23:929–947, 2002.

[5] Karen Braman, Ralph Byers, and Roy Mathias. The multi-shift QR algorithm Part II: Aggressive early deflation. *SIAM J. Matrix Anal. Appl.*, 23:948–973, 2002.

[6] David Day. How the shifted QR algorithm fails to converge and how to fix it. Technical Report 96-0913, Sandia National Labs, PO Box 5800, Albuquerque, NM 87185, 1996.

[7] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 16:1–17, 1990.

[8] J. G. F. Francis. The QR transformation: A unitary analogue to the LR transformation, parts I and II. *Comput. J.*, 4:265–272, 332–345, 1961.

[9] Gene H. Golub and Charles F. Van Loan. *Matrix computations*, volume 3 of *Johns Hopkins Series in the Mathematical Sciences*. Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

[10] Kazushige Goto and Robert van de Geijn. On reducing TLB misses in matrix multiplication, FLAME working note #9. Technical Report TR-2002-55, The University of Texas at Austin, Department of Computer Sciences, Taylor Hall 2.124, 1 University Station C0500, Austin, Texas 78712-0233, USA, 2002.

[11] Kazushige Goto and Robert A. van de Geijn. Anatomy of a high-performance matrix multiplication. *ACM Transactions on Mathematical Software*, to appear.

[12] D. Kressner. The effect of aggressive early deflation on the convergence of the QR algorithm. Technical Report Uminf report, Department of Computing Science, Umeøa University, Sweden, 2006.

[13] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Algorithm 539: Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Software*, 5:324–325, 1979.

[14] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Software*, 5:308–323, 1979.

[15] David S. Watkins. Forward stability and transmission of shifts in the QR algorithm. *SIAM J. Matrix Anal. Appl.*, 16:469–487, 1995.

[16] David S. Watkins. The transmission of shifts and shift blurring in the QR algorithm. *Linear Algebra Appl.*, 241/243:877–896, 1996.

[17] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Claredon Press, Oxford, UK, 1965.