The Computation of Elementary Unitary Matrices

R.B. Lehoucq*

April 20, 1994

Abstract

The construction of elementary unitary matrices that transform a complex vector to a real multiple of e_1, the first column of the identity matrix, are studied. We survey the two well known forms and present what appears to be relatively unknown third form implemented by LAPACK subroutine CLARFG.

1 Introduction

Define the elementary Hermitian (or unitary) matrix

$$U = I - 2ww^H, \tag{1}$$

where $w^H w = 1$. Easily verified is that U is both Hermitian and unitary and is also refered to as a Householder matrix. The matrix U as defined by (1) is a special case of the more general class of *elementary* matrices defined by

$$E(u, v; \sigma) = I - \sigma uv^H. \tag{2}$$

See [2], [3] and [4] for further details. The latter reference is a comprehensive study of elementary matrices, their properties and extensions to *block* implementations.

In [3] Wilkinson states that it is not possible, in general, to find an elementary matrix U such that

$$Ux = \alpha e_1, \tag{3}$$

for a complex vector x with α a real number. Note that U is unitary implies that $|\alpha| = \|x\|_2$. If the above relation were satisfied then

$$x^H \alpha e_1 = x^H U x.$$

The righthand side is a real number since U is Hermitian. But if $e_1^T x$ has a non-zero imaginary part we have a contradiction. The necessary condition is that $x^H e_1 = e_1^T x$. Equivalently $e_1^T x = \xi_1$ must have zero imaginary part.

The purpose of this note is to review and examine the case of constructing elementary unitary matrices that satisfy (3) for a complex vector x. The next section

*Department of Computational and Applied Mathematics, Rice University(lehoucq@rice.edu).
This work was supported by DARPA under contract TV-ORA4466.01

discusses the approach suggested by Wilkinson. The third section introduces yet another approach due to Hammarling and Du Croz. It has been implemented in the NAG Fortran Library subroutine F06HRF, see [1]. A slightly different implementation is the LAPACK subroutine CLARFG. The details of the software implementation for CLARFG are in section four. In fact what led to this study is attempting to understand the differences between the Wilkinson approach and the alternate formulation implemented by LAPACK, [6]. Section five generalizes the Hammarling–Du Croz form for transforming a complex vector x into another complex vector y. The last section contains the source code for subroutine CLARFG.

2 The Wilkinson Approach

Wilkinson suggests the following modification in Chapter 2, section 45 of [3]. If $e_1^T x = \xi_1$ has non-zero imaginary part then

$$\begin{aligned} x &= e^{i\theta_1}[|\xi_1|, e^{-i\theta_1}\xi_2, \ldots, e^{-i\theta_1}\xi_n]^T, \\ &= e^{i\theta_1} y, \end{aligned}$$

where $e^{i\theta_1}|\xi_1|$ is the polar form for ξ_1. Since $e_1^T y$ is a real number, a Householder matrix P may be constructed satisfying (3). Set $U = e^{-i\theta_1}P$ and

$$\begin{aligned} Ux &= e^{-i\theta_1}Px, \\ &= e^{-i\theta_1}Pe^{i\theta_1}y, \\ &= Py, \\ &= \alpha e_1. \end{aligned}$$

The matrix U is not Hermitian but is unitary which is the crucial property. The value of α is not affected;

$$\begin{aligned} |\alpha| &= \|Ux\|_2, \\ &= |e^{-i\theta_1}|\,\|Px\|_2, \\ &= \|x\|_2. \end{aligned}$$

The matrix U constructed is not an elementary matrix. If ξ_1 is real then θ_1 is arbitrary and may be chosen equal to zero. The resulting elementary matrix is then also Hermitian.

The Wilkinson approach is used by LINPACK when working with complex data.

3 An Alternate Approach

The final form for an elementary unitary matrix studied is due to Hammarling and Du Croz, see [1] (Introduction – F06). We now derive this alternate form. Consider the general form (2) for an elementary unitary matrix $U = E(u, v; \sigma)$. The matrix must be unitary from which it follows that

$$\begin{aligned} I &= U^H U, \\ &= (I - \sigma uv^H)^H(I - \sigma uv^H), \\ &= I - \bar{\sigma}vu^H - \sigma uv^H + \sigma\bar{\sigma}(u^H u)vv^H, \end{aligned}$$

and cancelling terms results in

$$\sigma\bar{\sigma}(u^H u)vv^H \quad = \quad \bar{\sigma}vu^H + \sigma uv^H. \tag{4}$$

Rearranging terms gives

$$(\sigma\bar{\sigma}(u^H u)v - \sigma u)v^H \quad = \quad \bar{\sigma}vu^H,$$

and a row space argument implies that u and v are linearly dependent. Substituting $u = v$ into (4) results in

$$\begin{aligned} |\sigma|^2\|v\|_2^2 &= \sigma + \bar{\sigma}, \qquad (5)\\ &= 2Re(\sigma), \end{aligned}$$

determining the required relationship between σ and v. Choosing $v = x - \alpha e_1$ we have

$$\begin{aligned} Ux &= (I - \sigma vv^H)x, \\ &= x - (\sigma v^H x)v, \\ &= \alpha e_1, \end{aligned}$$

if $\sigma^{-1} = v^H x$. This choice of σ will satisfy (5) as we now demonstrate. First

$$\begin{aligned} v^H x &= (x^H - \alpha e_1^T)x, \\ &= x^H x - \alpha\xi_1, \\ &= \alpha(\alpha - \xi_1), \end{aligned}$$

which determines σ and

$$\begin{aligned} \|v\|_2^2 &= v^H v, \\ &= (x^H - \alpha e_1^T)(x - \alpha e_1), \\ &= 2\alpha(\alpha - \mathrm{Re}(\xi_1)), \end{aligned}$$

follows. Finally

$$\begin{aligned} (v^H x)(x^H v)(\sigma + \bar{\sigma}) &= (v^H x)(x^H v)(\frac{1}{v^H x} + \frac{1}{x^H v}), \\ &= x^H v + v^H x, \\ &= \alpha(\alpha - \xi_1) + \alpha(\alpha - \bar{\xi}_1), \\ &= 2\alpha(\alpha - \mathrm{Re}(\xi_1)), \end{aligned}$$

shows

$$\frac{\sigma + \bar{\sigma}}{|\sigma|^2} \quad = \quad \|v\|_2^2,$$

as claimed. The form as outlined in this section does not appear to be as widely known as the Wilkinson one. It has the benefit of transforming a complex vector x directly to real multiple of e_1. When ξ_1 is purely real then U is Hermitian. Unlike the Wilkinson variant the Hammarling–Du Croz approach results in an elementary unitary matrix.

4 Software Implementation of CLARFG

The actual implementation of the Hammarling–Du Croz variant has some slight modifications. The resulting code is an excellent example of the art of developing software from a numerical algorithm. Subroutine CLARFG determines an elementary unitary matrix $U = I - \tau u u^H$ such that $U^H x = \alpha e_1$ where $|\alpha| = \|x\|_2$ for α a real number. Simplifying the implementation of other algorithms in LAPACK requiring the use of elementary unitary matrices the normalization $u^H e_1 = 1$ is taken. This normalization is done for storage considerations and is discussed in [5]. From the previous section it follows that

$$\begin{aligned} U^H &= I - \frac{1}{\alpha(\alpha - \bar{\xi}_1)} v v^H, \\ &= I - \frac{(\xi_1 - \alpha)(\bar{\xi}_1 - \alpha)}{\alpha(\alpha - \bar{\xi}_1)} u u^H, \\ &= I - \frac{\alpha - \xi_1}{\alpha} u u^H, \\ &= I - \tau u u^H, \end{aligned}$$

where

$$\begin{aligned} u &= \frac{v}{\xi_1 - \alpha}, \\ &= \frac{x - \alpha e_1}{\xi_1 - \alpha}, \end{aligned}$$

and

$$\tau = \frac{\alpha - \xi_1}{\alpha}.$$

Storage of U for use in further computation only requires storage for the complex τ and x may be overwritten with both α and the *essential* part of u, i.e.

$$x \leftarrow [\alpha, \frac{\xi_2}{\xi_1 - \alpha}, \ldots, \frac{\xi_n}{\xi_1 - \alpha}]^T.$$

In order that the value for τ and the scaling factor $\xi_1 - \alpha$ have small relative error

$$\alpha \leftarrow -\text{sign}(Re(\xi_1))\|x\|_2,$$

is chosen. This is a standard modification and is mentioned for the sake of completeness. The resulting τ satisfies the two properties

$$\begin{aligned} 1 \;\leq\; \text{Re}(\tau) \;&\leq\; 2, \\ |\tau - 1| \;&\leq\; 1, \end{aligned}$$

when $x \neq \gamma e_1$ with γ real. If x is a real multiple of e_1 then set

$$\begin{aligned} \tau &\leftarrow 0, \\ U &\leftarrow I. \end{aligned}$$

The reviewer of CLARFG will notice one final point which needs explanation. The careful programmer took care not to reciprocate the number $\|x\|_2$ that may fall below a certain machine dependent tolerance, SAFMIN. The value SAFMIN, computed by the LAPACK auxiliary subroutine SLAMCH is a machine dependent lower bound for numbers that may be safely reciprocated and not cause an overflow condition. If $\|x\|_2$ is less than the lower bound then the vector x is scaled by a multiple of the reciprocal of SAFMIN until $\|\theta x\|_2$ is at least as large as SAFMIN. Defining the integer k to represent the number of scalings required results in

$$\theta \quad = \quad k\frac{1}{\text{SAFMIN}}.$$

The number τ may now be safely computed as

$$\tau \quad \leftarrow \quad \frac{\|\theta x\|_2 - \theta\xi_1}{\|\theta x\|_2}.$$

In a similar fashion

$$\alpha \quad \leftarrow \quad -\text{sign}(Re(\xi_1))\frac{1}{\theta}(\|\theta x\|_2),$$

and the essential part of u

$$u \quad \leftarrow \quad \frac{1}{\theta\xi_1 - \theta\alpha}[\theta\xi_2, \ldots, \theta\xi_n]^T,$$

are computed.

5 Generalizations

We conclude with the problem of determining an elementary unitary matrix U that satisfies

$$\begin{aligned} Ux &= (I - \sigma v v^H)x, \\ &= y, \end{aligned} \tag{6}$$

where $\|x\|_2 = \|y\|_2$. The derivation of section three requires little modification. The value of $\sigma^{-1} = v^H x$ and

$$\begin{aligned} v &= x - y, \\ v^H x &= \|x\|_2^2 - y^H x, \end{aligned}$$

may be shown to satisfy (5) and (6).

6 Acknowledgements

The author would like to thank Jermey Du Croz for background information on the Hammarling–Du Croz approach and for encouragement. Both Jeremy Du Croz and Dan Sorensen carefully read preliminary manuscripts and their comments are appreciated. The clever scaling in section four is due to James Demmel.

References

[1] *NAG Fortran Library Manual, Mark 16*, Numerical Algorithms Group Ltd, Oxford, UK, 1993.

[2] A. H. Householder, *The Theory of Matrices in Numerical Analysis*, Blaisdell, 1964.

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

[4] A. A. Dubrulle, *Work Notes on Elementary Matrices*, Hewlett-Packard Laboratories, 1993, Technical Report HPL-93-69.

[5] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins, second edition, 1989.

[6] E. Anderson, Z. Bai, C. Bischoff, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen, *LAPACK Users' Guide*, SIAM, 1992.

[7] J. J. Dongarra, C. B. Moler, J. R. Bunch, G. W. Stewart, *LINPACK Users' Guide*, SIAM, 1979.

7 LAPACK Subroutine CLARFG

```
      SUBROUTINE CLARFG( N, ALPHA, X, INCX, TAU )
*
*  -- LAPACK auxiliary routine (version 1.1) --
*     Univ. of Tennessee, Univ. of California Berkeley, NAG Ltd.,
*     Courant Institute, Argonne National Lab, and Rice University
*     February 29, 1992
*
*     .. Scalar Arguments ..
      INTEGER            INCX, N
      COMPLEX            ALPHA, TAU
*     ..
*     .. Array Arguments ..
      COMPLEX            X( * )
*     ..
*
*  Purpose
*  =======
*
*  CLARFG generates a complex elementary reflector H of order n, such
*  that
*
*        H' * ( alpha ) = ( beta ),   H' * H = I.
*             (   x   )   (   0  )
*
```

```
*  where alpha and beta are scalars, with beta real, and x is an
*  (n-1)-element complex vector. H is represented in the form
*
*        H = I - tau * ( 1 ) * ( 1 v' ) ,
*                      ( v )
*
*  where tau is a complex scalar and v is a complex (n-1)-element
*  vector. Note that H is not hermitian.
*
*  If the elements of x are all zero and alpha is real, then tau = 0
*  and H is taken to be the unit matrix.
*
*  Otherwise  1 <= real(tau) <= 2  and  abs(tau-1) <= 1 .
*
*  Arguments
*  =========
*
*  N       (input) INTEGER
*          The order of the elementary reflector.
*
*  ALPHA   (input/output) COMPLEX
*          On entry, the value alpha.
*          On exit, it is overwritten with the value beta.
*
*  X       (input/output) COMPLEX array, dimension
*                         (1+(N-2)*abs(INCX))
*          On entry, the vector x.
*          On exit, it is overwritten with the vector v.
*
*  INCX    (input) INTEGER
*          The increment between elements of X. INCX <> 0.
*
*  TAU     (output) COMPLEX
*          The value tau.
*
*  =====================================================================
*
*     .. Parameters ..
      REAL               ONE, ZERO
      PARAMETER          ( ONE = 1.0E+0, ZERO = 0.0E+0 )
*     ..
*     .. Local Scalars ..
      INTEGER            J, KNT
      REAL               ALPHI, ALPHR, BETA, RSAFMN, SAFMIN, XNORM
*     ..
*     .. External Functions ..
      REAL               SCNRM2, SLAMCH, SLAPY3
```

```
      COMPLEX            CLADIV
      EXTERNAL           CLADIV, SCNRM2, SLAMCH, SLAPY3
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC          ABS, AIMAG, CMPLX, REAL, SIGN
*     ..
*     .. External Subroutines ..
      EXTERNAL           CSCAL, CSSCAL
*     ..
*     .. Executable Statements ..
*
      IF( N.LE.0 ) THEN
         TAU = ZERO
         RETURN
      END IF
*
      XNORM = SCNRM2( N-1, X, INCX )
      ALPHR = REAL( ALPHA )
      ALPHI = AIMAG( ALPHA )
*
      IF( XNORM.EQ.ZERO .AND. ALPHI.EQ.ZERO ) THEN
*
*        H  =  I
*
         TAU = ZERO
      ELSE
*
*        general case
*
         BETA = -SIGN( SLAPY3( ALPHR, ALPHI, XNORM ), ALPHR )
         SAFMIN = SLAMCH( 'S' )
         RSAFMN = ONE / SAFMIN
*
         IF( ABS( BETA ).LT.SAFMIN ) THEN
*
*           XNORM, BETA may be inaccurate; scale X and recompute them
*
            KNT = 0
   10       CONTINUE
            KNT = KNT + 1
            CALL CSSCAL( N-1, RSAFMN, X, INCX )
            BETA = BETA*RSAFMN
            ALPHI = ALPHI*RSAFMN
            ALPHR = ALPHR*RSAFMN
            IF( ABS( BETA ).LT.SAFMIN )
     $         GO TO 10
*
```

```
*           New BETA is at most 1, at least SAFMIN
*
            XNORM = SCNRM2( N-1, X, INCX )
            ALPHA = CMPLX( ALPHR, ALPHI )
            BETA = -SIGN( SLAPY3( ALPHR, ALPHI, XNORM ), ALPHR )
            TAU = CMPLX( ( BETA-ALPHR ) / BETA, -ALPHI / BETA )
            ALPHA = CLADIV( CMPLX( ONE ), ALPHA-BETA )
            CALL CSCAL( N-1, ALPHA, X, INCX )
*
*           If ALPHA is subnormal, it may lose relative accuracy
*
            ALPHA = BETA
            DO 20 J = 1, KNT
               ALPHA = ALPHA*SAFMIN
   20       CONTINUE
         ELSE
            TAU = CMPLX( ( BETA-ALPHR ) / BETA, -ALPHI / BETA )
            ALPHA = CLADIV( CMPLX( ONE ), ALPHA-BETA )
            CALL CSCAL( N-1, ALPHA, X, INCX )
            ALPHA = BETA
         END IF
      END IF
*
      RETURN
*
*     End of CLARFG
*
      END
```