Mathematical Programming Study 18 (1982) 102–111
North-Holland Publishing Company

USING THE HESSENBERG DECOMPOSITION IN CONTROL THEORY*

Charles VAN LOAN

Department of Computer Science, Cornell University, Ithaca, New York 14853, U.S.A.

Received 4 September 1980
Revised manuscript received 16 October 1981

Orthogonal matrix techniques are gaining wide acceptance in applied areas by practitioners who appreciate the value of reliable numerical software. Quality programs that can be used to compute the QR Decomposition, the Singular Value Decomposition, and the Schur Decomposition are primarily responsible for this increased appreciation. A fourth orthogonal matrix decomposition, the Hessenberg Decomposition, has recently been put to good use in certain control theory applications. We describe some of these applications and illustrate why this decomposition can frequently replace the much more costly decomposition of Schur.

Key words: Hessenberg Decomposition, Orthogonal Matrix.

1. Introduction

In the early part of this century, Schur proved that any n-by-n matrix A can be factored as

$$A = UTU^{H} \tag{1.1}$$

where U is unitary ($U^{H}U = I$) and T is upper triangular. This decomposition has an eminent role to play in numerical analysis because it is the 'output' of the well-known QR algorithm for eigenvalues [11].

Actually, in the typical case when A is real, the QR algorithm computes the 'Real Schur Decomposition'

$$A = VSV^{T} \tag{1.2}$$

where V is real orthogonal ($V^{T}V = I$) and S is block triangular having diagonal blocks that are either 1-by-1 (corresponding to real eigenvalues) or 2-by-2 (corresponding to complex conjugate eigenvalues). In this paper we assume that all matrices are real and consider (1.2) instead of (1.1).

Similar to the Real Schur Decomposition is the factorization

$$A = QHQ^{T} \tag{1.3}$$

where Q is orthogonal and H is (upper) Hessenberg, i.e., $h_{ij} = 0$ whenever $i > j + 1$. We refer to (1.3) as the 'Hessenberg Decomposition'. It has long been

* This work was partially supported by NSF Grant MCS 8004 106.

appreciated as a 'front end' factorization for the QR algorithm because the first step in computing (1.2) is to compute (1.3) and then apply the double implicit Francis iteration to H [11]. However, because the Hessenberg Decomposition has been successfully used to solve important applied problems, it has come to be recognized as a valuable tool in matrix computations in its own right, often as a cheaper alternative to (1.2).

In the sequel we describe some of these applications with the intention of increasing the reader's appreciation for this very useful orthogonal matrix factorization. But before we do this, we briefly describe in the next two sections the standard algorithm for computing (1.3) and the properties of Hessenberg matrices that can make them computationally attractive.

2. Computing the decomposition

The Hessenberg Decomposition is typically computed using Householder matrices. These are orthogonal matrices of the form

$$P = I_k - 2uu^T/u^Tu.$$

where u is a nonzero vector in $\mathbf{R}^k$ and I_k is the k-by-k identity matrix.

Householder matrices can be used to zero specified entries in a vector. Suppose x is a non-zero vector in $\mathbf{R}^k$ and define u by

$$u = x + \operatorname{sign}(x_1)\|x\|_2 e_1.$$

where e_1 is the first column of I_k. It then follows that

$$Px = -\operatorname{sign}(x_1)\|x\|_2 e_1,$$

a vector having zeroes in all but its first component. See Wilkinson [11] for more details.

Because of their ability to introduce zeros, Householder matrices can be used to compute the Hessenberg Decomposition. There are n-2 steps in the overall procedure. At the beginning of the jth step orthogonal matrices $P_1, \ldots, P_{j-1}$ have been found such that

$$A_{j-1} = (P_1 \cdots P_{j-1})^T A (P_1 \cdots P_{j-1})$$

has the form

$$A_{j-1} = \begin{bmatrix} H_{11}^{(j-1)} & \vdots & H_{12}^{(j-1)} \\ \cdots & \cdots & \cdots \\ 0 & b^{(j-1)} & H_{22}^{(j-1)} \end{bmatrix} \begin{matrix} j \\ \\ n-j \end{matrix}$$

$$\qquad\quad j-1 \quad 1 \quad n-j$$

where $H_{11}^{(j-1)}$ is Hessenberg. Let

$$\tilde{P}_j = I_{n-j} - 2u^{(j)}u^{(j)T}/u^{(j)T}u^{(j)}$$

be a Householder matrix with the property that $\tilde{P}_j b^{(j-1)}$ has zeroes in its last $n-j-1$ components. It then follows that the matrix $P_j = \text{diag}(I_j, \tilde{P}_j)$ is orthogonal and that $A_j \equiv P_j^T A_{j-1} P_j$ has the form

$$A_j = \begin{bmatrix} H_{11}^{(j)} & \vdots & H_{12}^{(j)} \\ \cdots & \vdots & \cdots \\ 0 & \vdots & H_{22}^{(j)} \end{bmatrix} \begin{matrix} j+1 \\ n-j+1 \end{matrix}$$

$$\begin{matrix} j & 1 & n-j+1 \end{matrix}$$

where $H_{11}^{(j)}$ is Hessenberg. This illustrates the jth step in the algorithm. It is clear that A_{j-2} is Hessenberg and that the matrix Q in (1.3) is given by $Q = P_1 \cdots P_{j-2}$.

A careful operation count reveals that $5n^3/3$ flops are required for the complete algorithm. (A 'flop' is the *approximate* amount of computer arithmetic necessary to perform a FORTRAN statement of the form $A(I,J) = A(I,J) - T * A(K,J)$.) This operation count assumes that Q is not explicitly formed but rather left in 'factored form'. That is, instead of storing Q in an n-by-n array, the Householder vectors $u^{(1)}, \ldots, u^{(n-2)}$ are stored. Then, whenever a calculation of the form $y = Qx$ has to be performed, x is premultiplied by the matrices $P_{n-2}, \ldots, P_1$ defined above. This can be accomplished in n^2 flops. An explicit copy of Q, if it is needed, can be formed in $2n^3/3$ flops.

The subroutines ORTHES and ORTRAN in EISPACK [9] are designed to perform the calculations that we have described above. ORTHES computes H and ORTRAN forms Q from the $n-2$ Householder vectors. As is typical of algorithms based on orthogonal transformations of data, the routines are very stable. For example, it can be shown that if H is the Hessenberg matrix produced by ORTHES on a computer having machine precision EPS, then

$$H = Q^T(A+E)Q$$

where

$$\|E\|_2 \leq c \text{ EPS } n^2 \|A\|_2$$

where Q is exactly orthogonal and c is some small constant. That is, H would result if ORTHES was applied in exact arithmetic to the 'nearby' matrix $A+E$. See Wilkinson [11] for a complete analysis.

The same desirable roundoff properties characterize the quasitriangular matrix produced by the QR algorithm. However, to compute (1.2), approximately $15n^3$ flops are required. This is about 6 times the work required to calculate the Hessenberg Decomposition. Thus, if a given problem can be solved by computing either (1.2) or (1.3), then the latter should normally be preferred.

We mention in passing that if one is willing to work with stabilized elementary transformations rather than orthogonal transformations, then A can be reduced to Hessenberg form via non-orthogonal similarity transformations in about $5n^3/6$ flops [11]. The EISPACK subroutines ELMHES AND ELMTRAN can be used for this purpose. However, as in the case of Gaussian elimination with partial

pivoting, there is the (remote) possibility of severe element growth [2]. Because of this and because of our wish to emphasize orthogonal matrix techniques, we will only consider the reduction to Hessenberg form via Householder matrices.

3. Important properties of Hessenberg matrices

In the applications that follow, there are primarily two properties of Hessenberg matrices that are exploited:

(i) Powering a Hessenberg matrix is cheap.

(ii) Solving a Hessenberg system of equations is cheap.

By 'cheap' we mean in comparison to the corresponding algorithms for general matrices. Property (i) is based on the fact that if H is Hessenberg and $B = H^k$, then b_{ij} is zero for all $i > j + k$. That is, H^k has lower bandwidth k. Because of this, approximately

$$\min\left\{\frac{n^3}{2}, \frac{n^3}{6} + \frac{k(n-k)(n+k)}{2} + \frac{k^3}{3}\right\}$$

flops are required to compute $C = H^{k+1}$ from $B = H^k$ as a flop count of the following algorithm indicates:

For $i = 1$ to n

For $j = \max\{1, i-k-1\}$ to n

$$c_{ij} = \sum_{p=\max\{1,i-1\}}^{\min\{j+k,n\}} h_{ip} b_{pj}$$

If $k \ll n$, then clearly each power of H costs about $n^3/6$ flops. This is essentially the same amount of work that is required for each power of a quasi-triangular matrix. In contrast, powers of a general n-by-n matrix require n^3 flops.

Economies can also be made when solving Hessenberg systems of equations. Suppose Gaussian elimination with partial pivoting is used to compute the factorization

$$PH = LU \tag{3.1}$$

where L is lower triangular with ones along the diagonal, U is upper triangular, and P is a permutation chosen so that each entry in L is bounded by one in modulus. It can be shown that only n^2 flops are required to compute (3.1) because there is only one row operation to perform in each of the $n-1$ steps of the elimination. This implies that the matrix L is essentially lower bidiagonal.

Once (3.1) is computed, a system of the form $Hx = b$ can be solved very quickly:

$Ly = Pb$ (n flops)

$Ux = y$ ($n^2/2$ flops).

Thus, a total of only n^2 flops are required to completely solve a Hessenberg linear system. This is only twice the work necessary to solve a quasi-triangular system, but more importantly, it is substantially less than the $n^3/3$ flops needed to solve a general n-by-n linear system.

4. Computing the matrix exponential

We now proceed to illustrate how the Hessenberg Decomposition can be used to speed certain basic algorithms that are frequently used in control theory. We begin by discussing the calculation of the matrix exponential.

One way to approximate the scalar function e^z is to use diagonal Padé approximants. These are rational functions of the form

$$r_{qq}(z) = \frac{s_{qq}(z)}{s_{qq}(-z)}$$

where

$$s_{qq}(z) = \sum_{k=0}^{q} \frac{(2q-k)!q!}{(2q)!k!(q-k)!} z^k.$$

If A is an n-by-n matrix, then

$$F_q(A) \equiv r_{qq}(A) = s_{qq}(-A)^{-1} s_{qq}(A)$$

is an approximation to the matrix exponential e^A. The quality of this approximation and the subtleties associated with using it in a practical algorithm are discussed in [7]. We mention only that $F_q(A)$ requires about qn^3 flops to evaluate. (There are $q-1$ matrix-matrix multiplications and a linear system to solve with n right hand sides.) The parameter q is typically between 4 and 8; its exact value depends upon the precision of the computer used and the accuracy required. For example, to attain full IBM 370 long precision accuracy, $q = 8$.

Suppose that we have computed the Hessenberg Decomposition $A = QHQ^T$. Since

$$F_q(A) = QF_q(H)Q^T \tag{4.1}$$

it follows that the evaluation of the Padé approximant will now involve powering a Hessenberg matrix and solving a linear system having lower bandwidth q. As a consequence, it can be shown that computing $F_q(A)$ via (4.1) requires only $(3 + q/6)n^3$ flops. If $q = 8$, then this is about half the work needed when $F_q(A)$ is calculated without any preliminary decomposition of A. In applications where $F_q(A)$ is only used to premultiply vectors, the amount of work can be even further reduced because then it is not necessary to multiply together the three matrices on the right in (4.1); Q can be accessed through its factored form. (The author is indebted to R.C. Ward [10] for calling his attention to these economies.)

We mention that there are many matrix function problems requiring the evaluation of a polynomial in a matrix. In these applications, the Hessenberg Decomposition can be used as above to save work. Of course, the Real Schur Decomposition can also be used to evaluate matrix polynomials:

$$A = VSV^{\mathrm{T}} \Rightarrow p(A) = Vp(S)V^{\mathrm{T}}.$$

However, although $p(S)$ is quasi-triangular and can be computed six times as fast as $p(A)$, the effective savings are negated by the $15n^3$ flops that are required to compute S in the first place.

5. Sylvester's equation

Bartels and Stewart [1] give an efficient algorithm for solving the Sylvester equation

$$AX + XB = C, \qquad A \in \mathbf{R}^{n\times n}, \quad B \in \mathbf{R}^{m\times m}, \quad C \in \mathbf{R}^{n\times m}.$$

Their algorithm begins by computing the Real Schur Decompositions

$$A = VSV^{\mathrm{T}}, \qquad B^{\mathrm{T}} = UTU^{\mathrm{T}}.$$

The original Sylvester equation then becomes

$$SY + YT^{\mathrm{T}} = D \tag{5.1}$$

where $Y = V^{\mathrm{T}}XU$ and $D = V^{\mathrm{T}}CU$. Let $Y = [y_1, \ldots, y_n]$ and $D = [d_1, \ldots, d_n]$ be column partitionings of Y and D respectively. Assume for clarity that A and B have real eigenvalues so that both S and T are triangular. By equating kth columns in (5.1) and then solving for y_k, we obtain the following:

$$(S + t_{kk}I)y_k = d_k - \sum_{i=k+1}^{n} t_{ki}y_i. \tag{5.2}$$

The matrix Y can be determined by solving these quasi-triangular systems for $k = n, n-1, \ldots, 1$. The unknown matrix X is then given by $X = VYU^{\mathrm{T}}$. In the overall process, most of the work is associated with the calculation of the Real Schur Decompositions of A and B.

Now suppose that A is only reduced to Hessenberg form, $A = QHQ^{\mathrm{T}}$. Just as above, this leads to a transformed Sylvester equation of the form $HY + YT^{\mathrm{T}} = D$ where $X = QYU^{\mathrm{T}}$ and $D = Q^{\mathrm{T}}CU$. The columns of Y can then be determined by solving the Hessenberg systems

$$(H + t_{kk}I)y_k = d_k - \sum_{i=k+1}^{n} t_{ki}y_i.$$

for $k = n, n-1, \ldots, 1$. This 'Hessenberg–Schur' algorithm is due to Golub, Nash and Van Loan [4] and it is substantially faster than the Bartels–Stewart al-

gorithm because it only requires the computation of a single Real Schur Decomposition. As an example of the savings to be made, if $n = 2m$ then the Hessenberg–Schur approach is twice as fast as the method of Bartels and Stewart.

We mention in passing that no one has yet been able to devise a 'Hessenberg' method for the Lyapunov equation $AX + XA^T = C = C^T$. The best procedure for this problem remains the Bartels–Stewart algorithm and it requires the calculation of A's Real Schur Decomposition.

6. Implicit methods for time-invariant initial value problems

Let A be an n-by-n matrix and consider the initial value problem

$$\dot{x} = Ax + f(t), \qquad x(t_0) = x_0.$$

If the 2nd order Adams–Moulton method is applied to this problem, we obtain the following difference scheme:

$$\left(I - \frac{h_k}{2} A\right)x_{k+1} = \left(I + \frac{h_k}{2} A\right)x_k + \frac{h_k}{2}[f(t_k) + f(t_{k+1})] \tag{6.1}$$

where $h_k = t_{k+1} - t_k$ and x_k is an approximation to $x(t_k)$. Notice that a system of linear equations must be solved at each time step. If the step length varies from iteration to iteration and if x_{k+1} is computed by applying Gaussian elimination to the matrix $(I - \frac{1}{2}h_kA)$, then about $n^3/3$ flops are required each step. However, if we initially compute the Hessenberg Decomposition $A = QHQ^T$, then (6.1) transforms to

$$\left(I - \frac{h_k}{2} H\right)y_{k+1} = \left(I + \frac{h_k}{2} H\right)y_k + \frac{h_k}{2} Q^T[f(t_k) + f(t_{k+1})] \tag{6.2}$$

where $x_k = Qy_k$. The key observation is that now, x_k can be computed in order n^2 flops regardless of whether the step length changes or not.

We refer the reader to [3] for a complete study of the Hessenberg Decomposition's role in solving initial value problems. Of course, (6.1) undergoes a similar transformation if we substitute A's Real Schur Decomposition. In this case, x_{k+1} is obtained via the solution of a quasi-triangular system. But again, the expense of computing (1.2) offsets the economies that can be made during the iteration.

7. Frequency response problems

In studying the frequency response of the system

$$\dot{x}(t) = Ax(t) + Bu(t),$$

$$y(t) = Cx(t) + Du(t), \qquad x(t_0) = x_0$$

it is often necessary to evaluate matrices of the form

$$K(w) = P^{\mathrm{T}}(jwI - A)^{-1}R \tag{7.1}$$

for many different values of the scalar w. Here $A \in \mathbf{R}^{n\times n}$, $P \in \mathbf{R}^{n\times m}$, $R \in \mathbf{R}^{n\times m}$ and $j^2 = -1$.

Notice that the calculation of $K(w)$ requires solving a linear system with $(jwI - A)$ as the matrix of coefficients. If Gaussian elimination is used for this purpose, then about $\frac{1}{3}n^3 + n^2m + nm^2$ flops are needed for each different value of w. The amount of computation would be considerably reduced if instead of having to compute a different matrix factorization for each w, we could get by with just a single factorization computed at the outset.

The Hessenberg Decomposition can be used in this context. Suppose $A = QHQ^{\mathrm{T}}$ is computed once and for all along with the matrices $P^{\mathrm{T}}Q$ and $Q^{\mathrm{T}}R$. It is clear that

$$K(w) = (P^{\mathrm{T}}Q)(jwI - H)^{-1}(Q^{\mathrm{T}}R)$$

can then be evaluated in only $\frac{1}{2}n^2m + nm^2$ flops. Typically, $K(w)$ must be evaluated for many different values of w, and so the Hessenberg Decomposition can save considerable time. Laub [5] discusses this procedure in detail and shows how complex arithmetic can be avoided even if w is complex.

8. Conclusions and warning

The above examples illustrate the power and utility of the Hessenberg Decomposition. Lack of space prevents us from describing other Hessenberg techniques such as Paige's controllability algorithm [8]. However, in all fairness it is important to call attention to some algorithms associated with Hessenberg matrices that are numerically dangerous. These algorithms invariably require the invertibility of H's subdiagonal elements, something that normally cannot be guaranteed. However, even when these subdiagonal entries are small, numerical difficulties are encountered.

A nice way to illustrate this point is to consider the reduction of a Hessenberg matrix to companion matrix form. It can be shown that if $H \in \mathbf{R}^{n\times n}$ is a Hessenberg matrix with nonzero subdiagonal entries, then there exists a nonsingular upper triangular matrix Z such that

$$Z^{-1}HZ = C = \begin{bmatrix} 0 & 0 & \cdots & & c_0 \\ 1 & 0 & \cdots & & c_1 \\ 0 & 1 & \cdots & & c_2 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & c_{n-1} \end{bmatrix}.$$

C is said to be a 'companion matrix', a canonical form that frequently arises in control theory. The matrix Z can be computed as the product of 'elementary' triangular matrices, $Z = Z_1 \cdots Z_{n(n-1)/2}$. For example, the matrix Z_1 has the form

$$Z_1 = \left[\begin{array}{cc|c} 1 & -h_{11}/h_{21} & 0 \\ 0 & 1/h_{21} & \\ \hline & 0 & I_{n-2} \end{array}\right].$$

The first step in the reduction to companion form involves computing the matrix $Z_1^{-1}HZ_1$ which has the form

$$\begin{bmatrix} 0 & * & * & \cdots & * \\ 1 & * & * & \cdots & * \\ & h_{21}h_{32} & * & & \vdots \\ & \ddots & & \ddots & \\ 0 & & & h_{n,n-1} & * \end{bmatrix}$$

where '$*$' denotes an arbitrary nonzero entry. Clearly, large roundoff errors will contaminate this update if h_{21} is small. A more complete analysis is given in [11, Chapter 6]. We merely present this computation as typical of several algorithms that appear in the literature that are of dubious numerical quality. Other examples may be found in [7].

We close by mentioning that if A is symmetric, then the matrix H in (1.3) is symmetric and tridiagonal. All of the algorithms that we have presented can be specialized in a straight-forward fashion to take advantage of this added structure.

References

[1] R. Bartels and G.W. Stewart, "A solution of the equation $AX + XB = C$", *Communications of the Association for Computing Machinery* 15 (1972) 820–826.

[2] P. Businger, "Reducing a matrix to Hessenberg form", *Mathematics of Computation* 23 (1969) 819–821.

[3] W. Enright, "On the efficient and reliable numerical solution of large linear systems of ODE's", *IEEE Transactions on Automatic Control*, AC-24 (1979) 905–908.

[4] G.H. Golub, S. Nash and C. Van Loan, "A Hessenberg–Schur method for the problem $AX + XB = C$", *IEEE Transactions on Automatic Control*, AC-24 (1979) 909–913.

[5] A. Laub, "Efficient multivariable frequency response computations", *IEEE Transactions on Automatic Control*, AC-26 (1981) 407–408.

[6] R.S. Martin and J.H. Wilkinson, "Similarity reduction of a general matrix to Hessenberg form", *Numerische Mathematik* 12 (1968) 349–368.

[7] C.B. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix", *SIAM Review* 20 (1978) 801–836.

[8] C. Paige, "Properties of numerical algorithms related to computing controllability", *IEEE Transactions on Automatic Control* AC-26 (1981) 130–139
[9] B.T. Smith, J. Boyle, B. Garbow, M. Ikebe, V. Klema, and C.B. Moler, *Matrix eigensystem routines-EISPACK guide* (Springer, New York, 1974).
[10] R.C. Ward, private communication.
[11] J.H. Wilkinson, *The algebraic eigenvalue problem* (Oxford University Press, New York, 1965).