IMA Journal of Numerical Analysis (2010) **30**, 887–897
doi:10.1093/imanum/drp018
Advance Access publication on August 21, 2009

Householder triangularization of a quasimatrix

LLOYD N. TREFETHEN†
Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK

[Received on 4 July 2008]

A standard algorithm for computing the QR factorization of a matrix A is Householder triangularization. Here this idea is generalized to the situation in which A is a quasimatrix, that is, a 'matrix' whose 'columns' are functions defined on an interval $[a, b]$. Applications are mentioned to quasimatrix least squares fitting, singular value decomposition and determination of ranks, norms and condition numbers, and numerical illustrations are presented using the chebfun system.

Keywords: Householder triangularization; QR factorization; chebfun; quasimatrix; singular value decomposition.

1. QR factorization and Householder triangularization

Let A be an $m \times n$ matrix, where $m \geqslant n$. A (reduced) QR factorization of A is a factorization

$$A = QR, \qquad \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} = \begin{bmatrix} q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \end{bmatrix} \begin{bmatrix} r & r & r & r \\ & r & r & r \\ & & r & r \\ & & & r \end{bmatrix}, \tag{1.1}$$

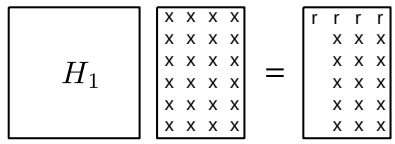
where Q is $m \times n$ with orthonormal columns and R is upper triangular. Here and in subsequent equations we illustrate our matrix operations by schematic pictures in which x denotes an arbitrary entry, r denotes an entry of an upper triangular matrix, q denotes an entry of an orthonormal column and a blank denotes a zero. QR factorization is a fundamental step for all kinds of computations in numerical linear algebra, including least squares, eigenvalues and singular value decomposition (SVD) (Björck, 1996; Golub & Van Loan, 1996; Trefethen & Bau, 1997; Stewart, 1998b).

One way to compute a QR factorization is by Gram–Schmidt or modified Gram–Schmidt factorization, a process of *triangular orthogonalization*. This is fast and straightforward, but the calculation may break down with division by zero if A is rank deficient, that is, if its column space has dimension less than n. Even if A has full rank, the matrix Q computed in floating point arithmetic may be far from orthogonal if A is ill conditioned, and this may have an adverse effect on the accuracy of subsequent computations (Stewart, 1998b).

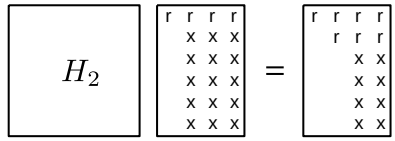
Householder (1958) introduced an alternative approach to (1.1) based on *orthogonal triangularization*, which has become one of the standard tools of computational science. At the first step a unitary

†Email: nick.trefethen@comlab.ox.ac.uk

matrix H_1 is applied that introduces zeros below the diagonal in the first column:

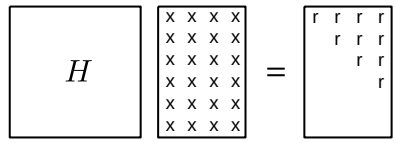


The second step applies another unitary matrix H_2 that leaves the first row unchanged and introduces zeros below the diagonal in the second column:



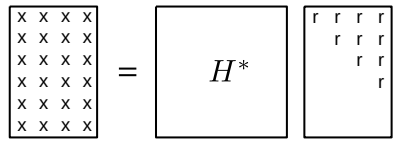
After n steps, A has become upper triangular:

$$H_n \cdots H_2 H_1 A = T, \qquad \text{(1.2)}$$



(If $m = n$ then the nth step is not needed. In these formulas we may take $H_n = I$.) Multiplying on the left by $H^* = H_1 H_2 \ldots H_n$ now gives

$$A = H_1 H_2 \cdots H_n T, \qquad \text{(1.3)}$$



We have written H_k instead of H_k^* since the two are the same because the Householder matrices H_k are self-adjoint as well as unitary. In fact, H_k is a *reflection matrix* that maps $\mathbb{C}^n$ to itself by reflection across an $(n-1)$-dimensional hyperplane. This interpretation makes it clear that $H_k^2 = I$, i.e., $H_k^{-1} = H_k$. Algebraically, H_k takes the form

$$H_k = I - 2v_k v_k^*, \qquad \text{(1.4)}$$

where v_k is a unit vector, i.e., $\|v_k\| = 1$. (Here * is the conjugate transpose and $\|\cdot\|$ is the 2 norm.) To multiply a Householder reflection by a vector or matrix, one always uses this form rather than an explicit entrywise representation, and that is why our schematic figures show H and H^* as boxes with no entries inside them.

For any unit vector v_k (1.4) defines a reflection in the sense just defined. In the application to QR factorization v_k is chosen orthogonal to the unit vectors $e_1, \ldots, e_{k-1}$, i.e., with zeros in positions $1, \ldots, k-1$, and this choice ensures that the space spanned by these vectors lies in the hyperplane that is invariant under H_k, which in turn ensures that zeros introduced at one step are not destroyed at later steps.

The factorization (1.3) is almost (1.1), but we are not quite there yet since the matrices H^* and T on the right have dimensions $m \times m$ and $m \times n$ rather than $m \times n$ and $n \times n$. To achieve the latter we need to delete all but the first n columns of H^* and all but the first n rows of T, which will have no effect on the product since the deleted rows are all zero. But if $H^* = H_1 \cdots H_n$ is never actually formed as a matrix then how do we delete some of its columns? We give an answer that will make the transition to quasimatrices particularly convenient as well as clarifying the treatment of matrices with complex entries. We shall factor T in (1.2) into a product of three matrices: an $m \times n$ 'rectangular identity' E whose columns are the unit vectors $e_1, \ldots, e_n$, an $n \times n$ 'sign matrix' $S = \mathrm{diag}(s_1, \ldots, s_n)$ whose diagonal entries are real or complex numbers with $|s_j| = 1$ and an $n \times n$ upper triangular matrix R whose diagonal entries are real and non-negative:

$$H_n \cdots H_2 H_1 A = ESR, \qquad \boxed{H} \boxed{\begin{matrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{matrix}} = \boxed{\begin{matrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{matrix}} \boxed{\begin{matrix} s & & & \\ & s & & \\ & & s & \\ & & & s \end{matrix}} \boxed{\begin{matrix} r & r & r & r \\ & r & r & r \\ & & r & r \\ & & & r \end{matrix}}. \tag{1.5}$$

This formulation makes it clear that we can compute the $m \times n$ matrix Q by applying $H_n, \ldots, H_1$ successively to ES:

$$Q = H_1 \cdots H_n ES, \qquad \boxed{\begin{matrix} q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \\ q & q & q & q \end{matrix}} = \boxed{H^*} \boxed{\begin{matrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{matrix}} \boxed{\begin{matrix} s & & & \\ & s & & \\ & & s & \\ & & & s \end{matrix}}. \tag{1.6}$$

2. Generalization to quasimatrices

Our subject in this note is the generalization of these ideas to the situation in which A is not a matrix but a *quasimatrix* defined on an interval $[a, b]$. By this we mean an '$\infty \times n$ matrix' (a more specific term would be '$[a, b] \times n$ matrix') whose 'columns' are functions of $x \in [a, b]$. Thus A represents a linear map from $\mathbb{C}^n$ to $L^2[a, b]$. The term quasimatrix comes from Stewart (1998a), and the same idea was introduced with different terminology in de Boor (1991) and Trefethen & Bau (1997, p. 52). We assume that each column lies in $L^2[a, b]$, so that inner products make sense, which we shall write in vector notation as $v^* w$. Then $\| \cdot \|$ is now the L^2 norm on $[a, b]$. For example, one might consider the quasimatrix with columns given by the functions $1, x, x^2, \ldots, x^{n-1}$ defined on $[-1, 1]$. The columns of Q in its QR factorization will be multiples of the Legendre polynomials $P_0, \ldots, P_{n-1}$, and the first few entries of R are $r_{11} = 2^{1/2}$, $r_{12} = 0$, $r_{22} = (2/3)^{1/2}$, $r_{13} = (2/9)^{1/2}$, $r_{23} = 0$ and $r_{33} = (8/45)^{1/2}$.

If A is a quasimatrix then the QR factorization formula (1.1) still makes sense. Now Q, like A, has dimensions $\infty \times n$, and it has orthonormal columns. In our schematic pictures a quasimatrix whose columns are functions is indicated by a rectangle containing vertical lines, with a 'q' in the middle if the functions are orthonormal:

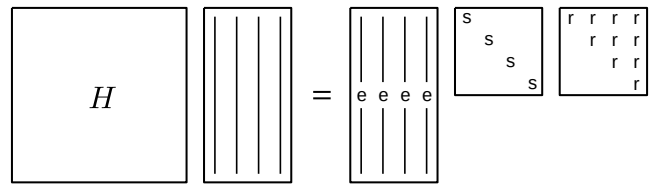
$$A = QR, \qquad \boxed{\begin{matrix} & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \end{matrix}} = \boxed{\begin{matrix} & & & \\ & & & \\ & & & \\ q & q & q & q \\ & & & \\ & & & \end{matrix}} \boxed{\begin{matrix} r & r & r & r \\ & r & r & r \\ & & r & r \\ & & & r \end{matrix}}. \tag{2.1}$$

A Gram–Schmidt computation goes exactly as before, with discrete inner products replaced by continuous ones, and is susceptible to the same difficulties if A is ill conditioned or rank deficient. Gram–Schmidt orthogonalization of quasimatrices was discussed in Battles & Trefethen (2004) and Battles (2006).

Another way to compute the QR decomposition of A would be to form A^*A, which is an $n \times n$ matrix, and then find its Cholesky factorization $A^*A = R^*R$ by standard methods. One then has $Q = AR^{-1}$ at least if A has full rank so that R is nonsingular. Like Gram–Schmidt orthogonalization, this method has difficulties if A is ill conditioned since it 'squares the condition number'. On the other hand, it provides an easy proof of a basic proposition: if the quasimatrix A has rank n and R has positive diagonal entries then the QR factorization is unique.

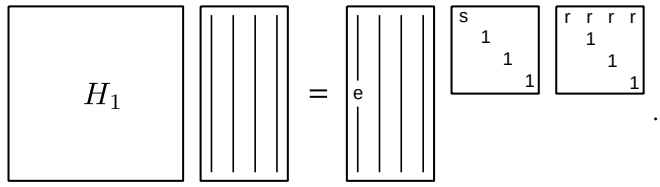
Our aim is to compute (2.1) by the numerically more stable method of Householder reflections, which has apparently not been done before. The very first step introduces the crucial question: What could it mean to map a function onto another function that is 'zero below the diagonal'? The answer we propose is that one should start from a quasimatrix analogue of (1.5):

$$H_n \cdots H_2 H_1 A = ESR, \qquad \text{(2.2)}$$



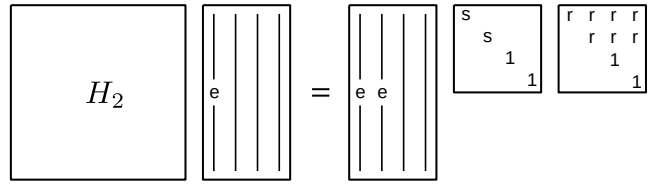
Here E will be an $\infty \times n$ quasimatrix, fixed in advance, with orthonormal columns $\{e_j\}$, and S will be an $n \times n$ sign matrix as before. There is no need for E to have any particular relationship to A, such as identical or similar column spaces. Just as in the matrix case, the Householder reflectors connecting A and E can reorient the vector space in arbitrary ways. In practice, we take e_j to be a multiple of the $(j-1)$th Legendre polynomial $P_{j-1}(x)$, scaled to $[a, b]$. The Householder reflector H_k will be a self-adjoint operator acting on $L^2[a, b]$, chosen so as to map a certain function to another function of equal norm in the space spanned by $e_1, \ldots, e_k$. Again, the signs $\{s_j\}$ will be chosen so that the diagonal entries of R are real and non-negative.

The first step of Householder quasimatrix triangularization looks like this:



A Householder reflector H_1 has been applied to A, changing all of its columns and, in particular, changing the first column to $r_{11}s_1e_1$, followed by making all of the remaining columns orthogonal to the first.

The second step looks like this:



The reflector H_2 has now changed the columns 2 to n, converting column 2 to $r_{12}s_1e_1 + r_{22}s_2e_2$. The process continues in this fashion until the form (2.2) is achieved at step n.

The precise formulas for these computations are as follows for each step k from 1 to n. The outer product notation vv^* denotes the operator that maps a function w to $v(v^*w)$. We have

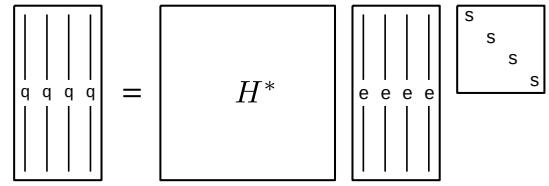
$$x = A(:,k),\ \ r_{kk} = \|x\|,\ \ s_k = -\text{sign}(e_k^* x),\ \ y = s_k r_{kk} e_k,\ \ v = \frac{y-x}{\|y-x\|},\quad H_k = I - 2vv^*, \tag{2.3}$$

$$A(:,j) = H_k A(:,j),\ \ r_{kj} = e_k^* A(:,j),\ \ A(:,j) = A(:,j) - r_{kj} e_k,\quad k+1 \leqslant j \leqslant n. \tag{2.4}$$

(The function sign(z) returns $z/\|z\|$ if $z \neq 0$, and 1 if $z = 0$.) The operations of (2.4) subtract off projections onto the current vector e_k from the remaining columns of A. The function x in (2.3) is accordingly orthogonal to $e_1, \ldots, e_{k-1}$, and this implies that the same property holds for v, which ensures that the application of H_k leaves the first $k-1$ columns of A unchanged.

After triangularization is completed, the computation of Q if it is needed goes as in (1.6):

$$Q = H_1 \cdots H_n \, E S, \tag{2.5}$$



The notation H^* makes sense since $H_1 \cdots H_n$ is indeed the adjoint of H in $L^2[a,b]$.

We now present our algorithm written in MATLAB-style pseudocode. The input is an $\infty \times n$ quasimatrix A, and the output is an $\infty \times n$ orthonormal quasimatrix Q and an $n \times n$ upper triangular matrix R with non-negative diagonal entries satisfying (2.1). One difference from the description above is that, instead of accumulating the signs s_k, we replace e_k by $s_k e_k$ at step k before moving on to step $k+1$. Another is that an additional line has been added to 'improve orthogonality' of the direction vector v_k to the previous target vectors $e_1, \ldots, e_{k-1}$. In exact arithmetic this would not be needed, but in floating point arithmetic we have found that it makes a significant difference.

HOUSEHOLDER TRIANGULARIZATION OF AN $\infty \times n$ QUASIMATRIX A

$E = \infty \times n$ *quasimatrix with orthonormal columns*
$V =$ *storage allocation for* $\infty \times n$ *quasimatrix*
$R =$ *storage allocation for* $n \times n$ *matrix*

```
for k = 1:n                                   TRIANGULARIZATION
    e = E(:,k)                                target for this reflection
    x = A(:,k)                                vector to be mapped to e
    ρ = ‖x‖,  R(k,k) = ρ
    α = e*x
    if α = 0, s = 1, else s = −α/‖α‖, end     compute real or complex sign
    e = se, E(:,k) = e                        modify e to take account of sign
    v = ρe − x                                vector defining reflection
    v = v − E(:,1:k−1)(E(:,1:k−1)*v)          improve orthogonality
    σ = ‖v‖
    if σ = 0, v = e, else v = v/σ, end        If zero column, arbitrary reflection
    V(:,k) = v                                store Householder vector
    J = (k+1:n)                               convenient abbreviation
    A(:,J) = A(:,J) − 2v(v*A(:,J))            apply reflection
    r^T = e*A(:,J)
    R(k,J) = r^T                              entries of row k of R
    A(:,J) = A(:,J) − er^T
end
Q = E
for k = n:−1:1                                FORMATION OF Q
    v = V(:,k)                                retrieve Householder vector
    J = (k:n)                                 convenient abbreviation
    Q(:,J) = Q(:,J) − 2v(v*Q(:,J))            apply reflection
end
```

3. Least squares and pseudoinverse

The most basic application of QR factorization is to the solution of least squares problems (Björck, 1996). The techniques used here carry over from matrices to quasimatrices with virtually no changes. Given a quasimatrix A of full rank and a function f, both defined on $[a, b]$, we seek a vector c such that

$$\|Ac - f\| = \text{minimum}. \tag{3.1}$$

As first worked out fully in Golub (1965), the solution can be obtained by solving the triangular system of equations

$$Rc = Q^* f, \tag{3.2}$$

with Q and R defined by (2.1). For this computation it is not necessary to form Q. It is enough if the Householder vectors v_k are stored and then used to compute $Q^* f$. One can also write (3.2) as $c = A^+ f$,

where A^+ is the *pseudoinverse* of A defined by

$$A^+ = R^{-1}Q^*. \tag{3.3}$$

The pseudoinverse is an $n \times \infty$ quasimatrix: it has n 'rows' each of which is a function on $[a, b]$.

4. SVD and related computations

Once the QR factorization (2.1) of a quasimatrix A is known, it is an easy matter to compute its SVD $A = U\Sigma V^*$. Following Battles & Trefethen (2004), we first compute the SVD $U_1\Sigma V^*$ of R, a standard problem since R is an ordinary matrix. We then set $U = QU_1$, giving

$$A = (QU_1)\Sigma V^* = U\Sigma V^*. \tag{4.1}$$

Here U is $\infty \times n$, Σ is $n \times n$ with diagonal form and nonincreasing non-negative diagonal entries $\{\sigma_j\}$ and V is $n \times n$ and unitary. This also gives us an alternative and numerically more stable formula for the pseudoinverse:

$$A^+ = V\Sigma^{-1}U^*. \tag{4.2}$$

As in the case of QR factorization, an alternative but less stable way to compute the SVD would be to compute A^*A and then work from the SVD of this $n \times n$ matrix.

From the SVD of A we can extract its norm

$$\|A\| = \sigma_1 \tag{4.3}$$

and its condition number

$$\kappa(A) = \sigma_1/\sigma_n, \tag{4.4}$$

where σ_1 and σ_n denote the largest and smallest singular values, respectively. Geometrically, these numbers have their usual interpretations as the lengths of the largest and smallest semiaxes of the n-dimensional hyperellipsoid that is the image under A of the unit ball in $\mathbb{C}^n$. The only difference from the matrix case is that this hyperellipsoid, while still itself finite dimensional, is now a subset of the infinite-dimensional space $L_2[a, b]$. If some of the singular values are zero then A is rank deficient, and its rank can be determined by counting the nonzero singular values. For a numerical rank determination algorithm one introduces a tolerance defining an approximate notion of 'nonzero'.

The method of computation of the SVD summarized in (4.1), making use of a preliminary QR factorization, is in the matrix case sometimes known as the 'Lawson–Hanson–Chan SVD' (Lawson & Hanson, 1974; Chan, 1982; Trefethen & Bau, 1997). For $m \times n$ matrices it is recommended in cases with $m > 5n/3$, so it seems natural that it should be appropriate in the quasimatrix case '$m = \infty$'. On the other hand, it is also possible to devise a quasimatrix SVD algorithm of the 'Golub–Kahan' variety in which one proceeds directly to bidiagonal form without a preliminary QR factorization (Golub & Kahan, 1965). This would involve n quasimatrix Householder reflections on the left, as we have used already, interleaved with $n - 1$ matrix Householder reflections on the right. If G^* denotes the product of these matrix Householder reflections then the form that results from Golub–Kahan bidiagonalization is

$$HA = EBG^*, \tag{4.5}$$

where B is an $n \times n$ upper bidiagonal matrix and G is an $n \times n$ unitary matrix of dimension $n \times n$. If $U_1 \Sigma V_1^*$ is an SVD of B then the SVD of A can be written as

$$A = (H^* E U_1) \Sigma (G V_1)^*. \tag{4.6}$$

5. Further algorithmic details

5.1 *Operation counts*

For matrix QR decomposition the Gram–Schmidt algorithm requires approximately $n^2/2$ inner products and approximately $n^2/2$ saxpys (scalar times vector plus vector). For $m \gg n$ this corresponds to a total operation count of approximately $2mn^2$ flops, where a flop is an addition, subtraction, multiplication or division. Householder triangularization costs the same if you only need R, but if Q must be formed as well then the operation counts double to approximately n^2 inner products and approximately n^2 saxpys.

In the quasimatrix case the Gram–Schmidt numbers are again approximately $n^2/2$ inner products and approximately $n^2/2$ saxpys (scalar times vector plus vector). Of course, in this context the meaning of an inner product or a saxpy is different since functions and quadrature are involved rather than discrete vectors. As for Householder triangularization of a quasimatrix, the second phase of forming Q again requires approximately $n^2/2$ inner products and approximately $n^2/2$ saxpys. The triangularization, however, is three times as expensive as before at least with the extra line to 'improve orthogonality', requiring approximately $3n^2/2$ inner products and approximately $3n^2/2$ saxpys. Thus, roughly speaking, whereas Householder QR factorization is twice as expensive as Gram–Schmidt for matrices, it is three or four times as expensive for quasimatrices. The reason is the extra orthogonalization operations needed at two points in the algorithm, which for matrices are bypassed since they are implied by the sparsity structure. We suspect it may be possible to improve these figures.

5.2 *Sparsity and Givens rotations*

For matrix QR factorization an alternative to Householder reflections is Givens rotations, which act on two rows at a time rather than $O(m)$ rows. Though more expensive when the matrix is dense, this technique may have advantages when the matrix is sparse. Now, what might it mean to say that a quasimatrix A is sparse? A possible answer would be that each column of A is orthogonal to most columns of E, assuming that E is specified in advance. Perhaps there may be problems where this property arises and analogues of Givens operations would be useful, but we have not pursued this idea.

6. Chebfun examples

This work was motivated by the chebfun software system, which makes it possible to compute with quasimatrices in MATLAB (Driscoll *et al.*, 2008). Chebfun version 1 used Gram–Schmidt orthogonalization, but beginning with Chebfun version 2.0308 in July 2008 we have replaced Gram–Schmidt orthogonalization by Householder triangularization in all QR factorizations. We finish with a few examples involving QR factorization and SVD computations in this system.

First, here is a computation of the norm and condition number of the set of functions $1, x, x^2, \ldots, x^5$ over $[-1, 1]$:

```
>> x = chebfun('x',[-1 1]);
```

```
>> A = [1 x x.^2 x.^3 x.^4 x.^5];
>> norm(A)
ans = 1.532062889375341
>> cond(A)
ans = 43.247975704139819
```

If $[-1, 1]$ is changed to $[0, 1]$ then the norm decreases to 1.272359956507724 and the condition number increases to 3866.659881620226.

Similarly, suppose that we want to know the dimension of the space spanned by the functions 1, $\sin^2(x)$ and $\cos^2(x)$. The following result comes out the same with either of the choices of domain of x mentioned above:

```
>> rank([1 sin(x).^2 cos(x).^2])
>> ans = 2
```

Here is an example involving piecewise smooth chebfuns. The commands

```
x = chebfun('x'); A = [];
for j = 0:6
   A = [A max(0,1-abs(3*(x+1)-j))];
end
```

construct an $\infty \times 7$ quasimatrix on $[-1, 1]$ whose columns correspond to triangular hat functions of width $1/3$ centred at $-1, -2/3, -1/3, \ldots, 1$. Linear combinations of these functions are piecewise linear functions on $[-1, 1]$ with breakpoints at $-2/3, -1/3, \ldots, 2/3$. The following code computes the least squares fit by such piecewise linear functions to $f(x) = e^x \sin(6x)$ and produces the plot shown in Fig. 1:

```
f = exp(x).*sin(6*x);
c = A\f;
```

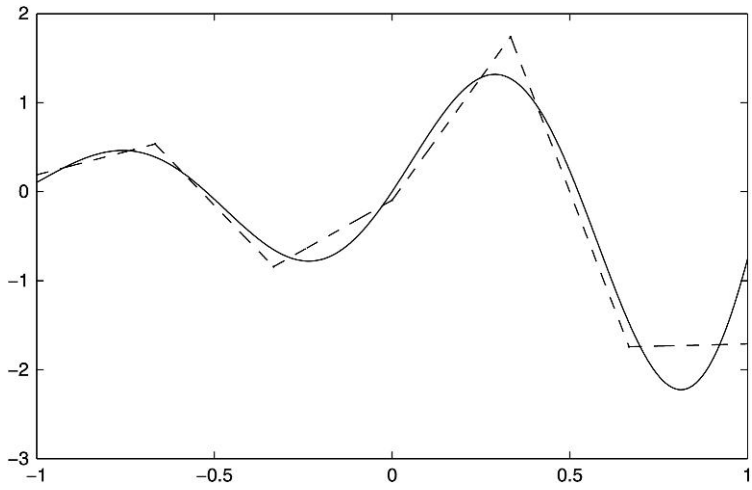


FIG. 1. Quasimatrix least squares fit by piecewise linear functions computed with the '\' command in the chebfun system. The residual norm minimized is defined by integrals, not point values. This quasimatrix has condition number 1.974212678743394.

```
ffit = A*c;
plot(f), hold on, plot(ffit,'--')
```

Here is the norm of the residual:

```
>> norm(f-ffit)
ans = 0.301000501411522
```

All of these computations are continuous, involving integrals, not just point values.

Finally, we recall that the advantage of Householder triangularization over Gram–Schmidt orthogonalization is that it delivers an orthonormal quasimatrix Q, even when A is rank deficient. We can verify this numerically by considering the highly rank-deficient quasimatrix consisting of two copies of A placed beside each other:

```
>> rank(A)
ans = 7
>> AA = [A A];
>> rank(AA)
ans = 7
>> [Q,R] = qr(AA);
>> cond(Q)
>> ans = 1.000000000000002
>> norm(AA-Q*R)
ans = 8.400509803176009e-16
```

The chebfun system is freely available, together with users guides and sample codes and other materials, at http://www.comlab.ox.ac.uk/chebfun.

7. Discussion

What is the essential algorithmic difference between factorizations of matrices and quasimatrices? Perhaps the main answer has to do with the notion of zero entries and associated matters of structures and sparsity. With ordinary matrices, the problem has already been formulated in the basis $e_1, e_2, \ldots$: to find out if a vector has zero component in a direction e_j we need only check if a certain number is zero. With quasimatrices, there is no basis given *a priori*. We must choose one to work with in practice—here we have taken suitably scaled Legendre polynomials—but the check of whether a function has zero component in a direction e_j now requires the computation of an inner product.

This note is a contribution to a larger project: to reconstruct the subject of numerical linear algebra in the context of functions rather than vectors. The project has an intellectual side, as it forces us to confront fundamental questions of the deeper meanings of the algorithms that we know and trust so well. It also has a practical side, a long-term enterprise indeed, which is to enable computational science and engineering to work directly with functions, with details of discretizations hidden away just as details of floating point arithmetic are today hidden away when we work with numbers.

Acknowledgements

This article was mostly written during the 17th Householder Symposium, held in June 2008 in Zeuthen, Germany. I am grateful for advice from several of the participants at the meeting and others including

Nick Higham, Jörg Liesen, Sasha Malyshev, Chris Paige, Beresford Parlett, Pete Stewart and Gil Strang. My chebfun teammates Toby Driscoll, Ricardo Pachón and Rodrigo Platte have also been very helpful.

Funding

EPSRC (EP/E045847/1).

References

BATTLES, Z. (2006) Numerical linear algebra for continuous functions. *D.Phil. Thesis*, Oxford University Computing Laboratory, Oxford.

BATTLES, Z. & TREFETHEN, L. N. (2004) An extension of MATLAB to continuous functions and operators. *SIAM J. Sci. Comput.*, **25**, 1743–1770.

BJÖRCK, Å (1996) *Numerical Methods for Least Squares Problems.* Philadelphia, PA: SIAM.

CHAN, T. F. (1982) An improved algorithm for computing the singular value decomposition. *ACM Trans. Math. Softw.*, **8**, 72–83.

DE BOOR, C. (1991) An alternative approach to (the teaching of) rank, basis, and dimension. *Linear Algebra Appl.*, **146**, 221–229.

DRISCOLL, T. A., PACHÓN, R., PLATTE, R. & TREFETHEN, L. N. (2008) *Chebfun Version 2*. Available at http://www.comlab.ox.ac.uk/chebfun/.

GOLUB, G. H. (1965) Numerical methods for solving least squares problems. *Numer. Math.*, **7**, 206–216.

GOLUB, G. H. & KAHAN, W. (1965) Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal.*, **2**, 205–224.

GOLUB, G. H. & VAN LOAN, C. (1996) *Matrix Computations*, 3rd edn. Baltimore, MD: Johns Hopkins University Press.

HOUSEHOLDER, A. S. (1958) Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.*, **5**, 339–342.

LAWSON, C. L. & HANSON, R. J. (1974) *Solving Least Squares Problems.* Englewood Cliffs, NJ: Prentice Hall.

STEWART, G. W. (1998a) *Afternotes Goes to Graduate School.* Philadelphia, PA: SIAM.

STEWART, G. W. (1998b) *Matrix Algorithms. Volume I: Basic Decompositions*. Philadelphia, PA: SIAM.

TREFETHEN, L. N. & BAU III, D. (1997) *Numerical Linear Algebra*. Philadelphia, PA: SIAM.