12-1990

HOUSEHOLDER REDUCTION

Per Brinch Hansen
Syracuse University, School of Computer and Information Science, pbh@top.cis.syr.edu

SU-CIS-90-39

HOUSEHOLDER REDUCTION

PER BRINCH HANSEN

December 1990

School of Computer and Information Science
Suite 4-116
Center for Science and Technology
Syracuse, New York 13244-4100

(315) 443-2368

HOUSEHOLDER REDUCTION[1]

PER BRINCH HANSEN

School of Computer and Information Science
Syracuse University
Syracuse, New York 13244

December 1990

Abstract. This tutorial discusses Householder reduction of n linear equations to a triangular form which can be solved by back substitution. The main strengths of the method are its numerical stability and suitability for parallel computing. We explain how Householder reduction can be derived from elementary matrix algebra. The method is illustrated by a numerical example and a Pascal algorithm. We assume that the reader has a general knowledge of vector and matrix algebra but is less familiar with linear transformation of a vector space.

Key words. Linear systems, Householder reduction.

AMS (MOS) subject classifications. 65F05 (direct methods for linear systems, Householder reduction).

Introduction. The solution of linear equations is important in many areas of science and engineering (Kreyszig [1988]). This tutorial discusses Householder reduction of n linear equations to a triangular form which can be solved by back substitution (Householder [1958], Press [1989]). The main strengths of the method are its numerical stability and suitability for parallel computing (Ortega [1988], Brinch Hansen [1990]). Text books on numerical analysis often produce Householder reduction like a rabbit from a magician's top hat. We will explain how the method can be derived from elementary matrix algebra. The method is illustrated by a numerical example and a Pascal algorithm.

We assume that the reader has a general knowledge of vector and matrix algebra but is less familiar with linear transformation of a vector space.

We begin by looking at the problems of Gaussian elimination.

1. Gaussian elimination. The classical method for solving a system of linear equations is Gaussian elimination. Suppose we have three linear equations with three unknowns $x_1,\ x_2, x_3$:

$$\begin{array}{rcrcrcr} 2x_1 & + & 2x_2 & + & 4x_3 & = & 18 \\ x_1 & + & 3x_2 & - & 2x_3 & = & 1 \\ 3x_1 & + & x_2 & + & 3x_3 & = & 14 \end{array}$$

First we eliminate x_1 from the second equation by subtracting 1/2 of the first equation from the second one. Then we eliminate x_1 from the third equation by subtracting 3/2 of the first equation from the third one. We now have three equations in which x_1 occurs in the first equation only

$$\begin{array}{rcrcrcr} 2x_1 & + & 2x_2 & + & 4x_3 & = & 18 \\ & & 2x_2 & - & 4x_3 & = & -8 \\ & - & 2x_2 & - & 3x_3 & = & -13 \end{array}$$

Finally we eliminate x_2 from the third equation by adding the second equation to the third one. The equations have now been reduced to a triangular form which has the same solution as the original equations but is easier to solve

$$\begin{array}{rcrcrcr} 2x_1 & + & 2x_2 & + & 4x_3 & = & 18 \\ & & 2x_2 & - & 4x_3 & = & -8 \\ & & & - & 7x_3 & = & -21 \end{array}$$

The triangular equations are solved by back substitution. From the third equation we immediately have $x_3 = 3$. By substituting this value in the second equation, we find $x_2 = 2$. Substituting these two values in the first equation we obtain $x_1 = 1$.

In general we have n linear equations with n unknowns

$$\begin{array}{rl} a_{11}x_1 + a_{12}x_2 + \ \cdots & + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ \cdots & + a_{2n}x_n = b_2 \\ \cdots & \\ a_{n1}x_1 + a_{n2}x_2 + \ \cdots & + a_{nn}x_n = b_n \end{array} \tag{1.1}$$

The a's and b's are known real numbers. The x's are the unknowns we must find.

The equation system (1.1) can be expressed as a vector equation

$$Ax = b \tag{1.2}$$

where A is an $n \times n$ matrix, while x and b are n-dimensional column vectors

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ \cdots \\ x_n \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ \cdots \\ b_n \end{bmatrix}$$

The equation system has a unique solution only if the matrix A is non-singular as defined in the appendix.

Gaussian elimination reduces Eq. (1.2) to an equivalent form

$$Ux = c$$

where U is an $n \times n$ upper triangular matrix

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

with all zeros below the main diagonal, while c is an n-dimensional column vector.

Gaussian elimination requires $0(n^3)$ operations.

The scaling of equations is a source of numerical errors in Gaussian elimination. To eliminate the i^{th} unknown from the j^{th} equation we subtract the i^{th} equation multiplied by a_{ji}/a_{ii} from the j^{th} equation. If the pivot element a_{ii} is very small, the scaling factor becomes very large and we may end up subtracting very large reals from very small ones. This makes the results highly inaccurate. The numerical instability of Gaussian elimination can be reduced by pivoting, a rearrangement of the rows and columns which makes the pivot element as large as possible.

On a parallel computer pivoting complicates the elimination algorithm (Fox [1988]). In the following we describe an alternative method which is numerically stable and does not require pivoting.

2. Scalar products. Householder reduction of an $n \times n$ real matrix has a simple geometric interpretation: The matrix columns are regarded as vectors in an n-dimensional space. Each vector is replaced by its mirror image on the other side of a particular plane. This plane reflects the first column onto the first axis of the coordinate system to produce a new column with all zeros after the first element.

Householder's method requires the computation of scalar products and vector reflections. The following is a brief explanation of these basic operations. The appendix defines the elementary laws of vector and matrix algebra, which we will take for granted.

Let a and b be two n-dimensional column vectors

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ \dots \\ a_n \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ \dots \\ b_n \end{bmatrix}$$

The transpose of a and b are the row vectors

$$a^T = [a_1\ a_2\ \dots\ a_n] \qquad b^T = [b_1\ b_2\ \dots\ b_n]$$

The scalar product of a and b is

(2.1) $$a^T b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

A scalar product is obviously symmetric

(2.2) $$a^T b = b^T a$$

The Euclidean norm

(2.3) $$\|a\| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$$

is the length of an n-dimensional vector a.

From Eqs. (2.1) and (2.3) we obtain an equivalent definition of the norm

(2.4) $$\|a\|^2 = a^T a$$

3. Reflection. Figure 1 shows a unit vector v in 3-dimensional space. The dotted line represents a plane P which is perpendicular to v through the origin O. For an arbitrary vector a we wish to find another vector b, which is the reflection of a on the other side of the plane P.

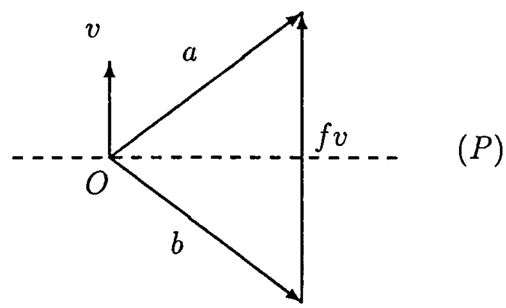

Fig. 1 Reflection.

The concept of reflection is defined by three equations:

The reflection plane P is determined by a vector v of length 1

(3.1) $$\|v\| = 1$$

Reflection preserves the norm of a vector

(3.2) $$\|a\| = \|b\|$$

The difference between a vector a and its reflection b is a vector fv which is a multiple of v

(3.3) $$fv = a - b$$

The (unknown) scalar f is the distance between the vector and its reflection.

We must find the reflection of an arbitrary vector a through a plane P defined by a given unit vector v.

$$\begin{aligned}
\|a\|^2 &= \|b\|^2 && \text{by (3.2)} \\
&= (a - fv)^T(a - fv) && \text{by (2.4), (3.3)} \\
&= a^Ta - fa^Tv - fv^Ta + f^2v^Tv \\
&= \|a\|^2 - 2fv^Ta + f^2 && \text{by (2.2), (2.4), (3.1)}
\end{aligned}$$

This equality determines the distance f between vector a and its image b

$$f = 2v^Ta \tag{3.4}$$

The reflection of b into a displaces b by the same distance f in the opposite direction. So we can also express the distance as

$$f = -2v^Tb \tag{3.5}$$

Finally we define b in terms of a and v

$$\begin{aligned}
b &= a - vf && \text{by (3.3)} \\
&= Ia - v(2v^Ta) && \text{by (3.4)} \\
&= (I - 2vv^T)a
\end{aligned}$$

where I is the $n \times n$ identity matrix defined in the appendix.

In other words, the reflection of a vector a is the vector

$$b = Ha \tag{3.6}$$

obtained by multiplying a by the $n \times n$ reflection matrix

$$H = I - 2vv^T \tag{3.7}$$

H is also called a Householder matrix. This is the rabbit that is often pulled out of the hat without any explanation of why it has this particular form.

Figure 1 is a geometric definition of reflection in 3-dimensional space. However, the algebraic equations derived from this figure make no assumptions about the dimension of space. In the following, we will simply say that Eqs. (3.6) and (3.7) define a transformation of an n-dimensional vector. By analogy we will call this transformation a "reflection" through an $(n-1)$-dimensional plane. The essential property is that reflection of an n-dimensional vector preserves the norm

$$\|Ha\| = \|a\| \tag{3.8}$$

This follows from Eqs. (3.2) and (3.6).

If we reflect a vector twice through the same plane, we get the same vector again

$$H(Ha) = a$$

In other words, two reflections are equivalent to an identity transformation

$$HH = I$$

Consequently H is a non-singular matrix which is its own inverse

$$H^{-1} = H$$

(see the appendix).

4. Householder reduction. We are looking for an algorithm that reduces an $n \times n$ real matrix A to triangular form without increasing the magnitude of the elements significantly.

An element of a column can never exceed the total length of the column vector. That is

$$|a_{ij}| \leq \|a_i\| \text{ for } i, j = 1, 2, \ldots, n$$

In other words, the norm of a column vector is an upper bound on the magnitude of its elements.

A method that changes the elements of a matrix A without changing the norms of its columns will obviously limit the magnitude of the matrix elements. This can be achieved by multiplying A by a Householder matrix H.

If we multiply a system of linear equations

$$Ax = b$$

by a non-singular matrix H, we obtain an equation

$$(HA)x = Hb$$

that has the same solution as the original system.

The first step in Householder reduction produces a matrix HA that has all zeros below the first element of the first column.

The reflection must transform column

$$a_1 = [a_{11}\ a_{21}\ \ldots\ a_{n1}]^T \tag{4.1}$$

into a column of the form

$$Ha_1 = [d_{11}\ 0\ \ldots\ 0]^T \tag{4.2}$$

where the diagonal element is

(4.3) $$d_{11} = \pm \|a_1\|$$

The choice of sign will be made later.

Equations (4.1)–(4.3) define the computation of the first column of the matrix HA.

The difference between column a_1 and its reflection Ha_1 is the column vector

$$\begin{aligned} f_1 v &= a_1 - b_1 && \text{by (3.3)} \\ &= a_1 - Ha_1 && \text{by (3.6)} \end{aligned}$$

Combining this with Eqs. (4.1) and (4.2) we find

(4.4) $$f_1 v = [w_{11}\ a_{21}\ \ldots\ a_{n1}]^T$$

where the first element is

(4.5) $$w_{11} = a_{11} - d_{11}$$

The distance between a_1 and its image Ha_1 is f_1 where

$$\begin{aligned} f_1^2 &= f_1(-2v^T Ha_1) && \text{by (3.5), (3.6)} \\ &= -2(f_1 v)^T Ha_1 \\ &= -2w_{11}\, d_{11} && \text{by (2.1), (4.2), (4.4)} \end{aligned}$$

In short

(4.6) $$f_1 = \sqrt{-2w_{11}\, d_{11}}$$

The unit vector v which determines the appropriate Householder matrix is

$$v = f_1 v / f_1$$

or by Eq. (4.4)

(4.7) $$v = [w_{11}\ a_{21}\ \ldots\ a_{n1}]^T / f_1$$

After the transformation of the first column a_1, each remaining column a_i is also replaced by its reflection through the same plane defined by Eqs. (3.3), (3.4), and (3.6).

(4.8) $$Ha_i = a_i - f_i v$$

$$f_i = 2v^T a_i \tag{4.9}$$

The reflection of a column is obtained by subtracting a multiple of the unit vector v.

5. Numerical stability. We still need to decide which sign to use for the diagonal element d_{11} in Eq. (4.3).

If $d_{11} = a_{11}$, the scalars w_{11} and f_1 are zero by Eqs. (4.5) and (4.6), and the division by f_1 in Eq. (4.7) causes overflow. We can avoid this problem by selecting the sign which makes $d_{11} \neq a_{11}$.

The overflow problem occurs when a_1 is a multiple of the unit vector

$$e_1 = [1\ 0\ \ldots\ 0]^T$$

For $a = a_{11}e_1$ there are four cases to consider

$$\begin{array}{lrcrl}
a_{11} > 0: & & & & \\
& d_{11} = +\,\|a_1\| & = & a_{11} & \text{(overflow)} \\
& d_{11} = -\,\|a_1\| & = & -a_{11} & \text{(no overflow)} \\
a_{11} < 0: & & & & \\
& d_{11} = +\,\|a_1\| & = & -a_{11} & \text{(no overflow)} \\
& d_{11} = -\,\|a_1\| & = & a_{11} & \text{(overflow)}
\end{array}$$

If a_1 is close to a multiple of e_1, serious rounding errors may occur if f_1 is very small.

This insight leads to the following rule

$$d_{11} = \text{if } a_{11} > 0 \text{ then } -\|a_1\| \text{ else } \|a_1\| \tag{5.1}$$

6. Computational rules. We are now ready to summarize the rules for computing the matrix HA as defined by Eqs. (2.1), (4.2), (4.5)–(4.9), and (5.1):

$$\begin{aligned}
\|a_1\| &= \sqrt{a_1^T a_1} \\
d_{11} &= \text{if } a_{11} > 0 \text{ then } -\|a_1\| \text{ else } \|a_1\| \\
w_{11} &= a_{11} - d_{11} \\
f_1 &= \sqrt{-2w_{11}d_{11}} \\
Ha_1 &= [d_{11}\ 0\ \ldots\ 0]^T \\
v &= [w_{11}\ a_{21}\ \ldots\ a_{n1}]^T / f_1 \\
f_i &= 2v^T a_i \quad \text{for } 1 < i \leq n \\
Ha_i &= a_i - f_i v
\end{aligned} \tag{6.1}$$

Householder's algorithm reduces a system of linear equations to upper triangular form in $n - 1$ steps:

The first step reduces A to a matrix HA with all zeros below the diagonal element in the first column. At the same time, b is transformed into a vector Hb. This computation, defined by Eq. (6.1), is called a Householder transformation.

$$
\begin{array}{cc}
HA & Hb \\
\left[\begin{array}{cc}
* & \begin{array}{ccc} * & \cdots & * \end{array} \\
\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \end{array} & \left[\begin{array}{ccc} * & \cdots & * \\ * & \cdots & * \\ \vdots & \vdots & \vdots \\ * & \cdots & * \end{array}\right]
\end{array}\right]
&
\left[\begin{array}{c} * \\ \left[\begin{array}{c} * \\ * \\ \vdots \\ * \end{array}\right] \end{array}\right]
\end{array}
$$

The second step reduces the $(n-1) \times (n-1)$ submatrix of HA shown above by Householder transformation. We now obtain a matrix with zeros below the diagonal elements in the first two columns. The same transformation is applied to the $(n-1)\times 1$ subvector of Hb shown above.

By a series of Householder transformations, applied to smaller and smaller submatrices and subvectors, the equation system is reduced, one column at a time, to upper triangular form.

7. A numerical example. We now return to the previous example of three equations with three unknowns. For convenience we combine the matrix A and the vector b into a single 3×4 matrix

$$
A0 = \begin{bmatrix} 2 & 2 & 4 & 18 \\ 1 & 3 & -2 & 1 \\ 3 & 1 & 3 & 14 \end{bmatrix}
$$

First we reduce $A0$ to a matrix $A1$ with all zeros below the diagonal element in the first column. This is done column by column using Eq. (6.1). The numbers shown below were produced by a computer and rounded to four decimal places.

First column:

$$
\begin{aligned}
a_1 &= [2\ 1\ 3]^T \\
v &= [0.8759\ 0.1526\ 0.4577]^T \\
f_1 &= 6.5549 \\
Ha_1 &= [-3.7417\ 0\ 0]^T
\end{aligned}
$$

Second column:

$$
\begin{aligned}
a_2 &= [2\ 3\ 1]^T \\
f_2 &= 5.3344 \\
Ha_2 &= [-2.6726\ 2.1862\quad -1.4414]^T
\end{aligned}
$$

Third column:

$$
\begin{aligned}
a_3 &= [4\ \ -2\ 3]^T \\
f_3 &= 9.1433 \\
Ha_3 &= [-4.0089\quad -3.3949\quad -1.1846]^T
\end{aligned}
$$

Fourth column:

$$
\begin{aligned}
a_4 &= [18\ 1\ 14]^T \\
f_4 &= 44.6536 \\
Ha_4 &= [-21.1136\quad -5.8123\quad -6.4368]^T
\end{aligned}
$$

We now have the matrix

$$A1 = \begin{bmatrix} -3.7417 & -2.6726 & -4.0089 & -21.1136 \\ 0 & 2.1862 & -3.3949 & -5.8123 \\ 0 & -1.4414 & -1.1846 & -6.4368 \end{bmatrix}$$

The next step of the algorithm reduces the 2×2 submatrix

$$A1' = \begin{bmatrix} 2.1862 & -3.3949 & -5.8123 \\ -1.4414 & -1.1846 & -6.4368 \end{bmatrix}$$

to

$$A2' = \begin{bmatrix} -2.6186 & 2.1822 & 1.3093 \\ 0 & -2.8577 & -8.5732 \end{bmatrix}$$

The final triangular matrix

$$A2 = \begin{bmatrix} -3.7417 & -2.6726 & -4.0089 & -21.1136 \\ 0 & -2.6186 & 2.1822 & 1.3093 \\ 0 & 0 & -2.8577 & -8.5732 \end{bmatrix}$$

consists of the first row and column of $A1$ and the submatrix $A2'$.

The triangular equation system is solved by back substitution to obtain

$$x = [1.0000 \quad 2.0000 \quad 3.0000]^T$$

8. Pascal algorithm. The following Pascal algorithm assumes that the matrix A is stored by columns, that is, $a[i]$ denotes the i^{th} column of A. For each submatrix of A, the eliminate operation is applied to the first column, and the transform operation is applied to each remaining column (including b).

```
type
  column = array [1..n] of real;
  matrix = array [1..n] of column;

procedure reduce (var a: matrix;
  var b: column);
var vi: column; i, j: integer;

  function product(i: integer;
    var a, b: column): real;
  { the scalar product of
    elements i..n of a and b }
  var ab: real; k: integer;
  begin
     ab := 0.0;
    for k := i to n do
       ab := ab + a[k]*b[k];
    product := ab
  end;
```

```
    procedure eliminate(i: integer;
      var ai, vi: column);
    var anorm, dii, fi, wii: real;
      k: integer;
    begin
      anorm :=
        sqrt(product(i, ai, ai));
      if ai[i] > 0.0
        then dii := -anorm
        else dii := anorm;
      wii := ai[i] - dii;
      fi := sqrt(-2.0*wii*dii);
      vi[i] := wii/fi;
      ai[i] := dii;
      for k := i + 1 to n do
      begin
        vi[k] := ai[k]/fi ;
        ai[k] := 0.0
      end
    end;

    procedure transform(i: integer;
      var aj, vi: column);
    var fi: real; k: integer;
    begin
      fi := 2.0*product(i, vi, aj);
      for k := i to n do
        aj[k] := aj[k] - fi*vi[k]
    end;

  begin
    for i := 1 to n - 1 do
    begin
      eliminate (i, a[i], vi);
      for j := i + 1 to n do
        transform(i, a[j], vi);
      transform(i, b, vi);
    end
  end
```

The execution time of the algorithm is $0(n^3)$.

Final remarks. We have explained Householder's method for reducing a matrix to triangular form. The main advantage of the method is that it achieves numerical stability without pivoting. We have illustrated the computation by an example and a

Pascal algorithm. Householder reduction is an interesting example of a fundamental computation with a subtle theory and a short algorithm.

Acknowledgements. To Fred Schlereth for bringing Householder's method to my attention, Nawal Copty for explaining it to me, and to Jonathan Greenfield and Erik Hemmingsen for advice on how to improve the presentation.

REFERENCES

E. KREYSZIG [1988], *Advanced Engineering Mathematics*, John Wiley & Sons, New York.

A. S. HOUSEHOLDER [1958], "Unitary triangularization of a nonsymmetric matrix," *J. ACM*, 5, pp. 339–342.

W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, and W. T. VETTERLING [1989], *Numerical Recipes in Pascal—The Art of Scientific Computing*, Cambridge University Press, New York.

J. M. ORTEGA [1988], *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York.

P. BRINCH HANSEN [1990], *The All-Pairs Pipeline*, School of Computer and Information Science, Syracuse University, Syracuse, New York, 1990.

G. C. FOX, M. A. JOHNSON, G. A. LYZENGA, S. W. OTTO, J. K. SALMON, and D. W. WALKER [1988], *Solving Problems on Concurrent Processors, Vol. I*, Prentice-Hall, Englewood Cliffs, New Jersey.

APPENDIX: MATRIX ALGEBRA

In the algebraic laws, A, B, and C denote matrices, while k is a scalar.
The identity matrix is

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ & & \dots & & \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

The transpose A^T is the matrix obtained by exchanging the rows and columns of the matrix A.

The inverse of a matrix A is a matrix A^{-1} such that

$$AA^{-1} = I$$

If A^{-1} exists then A is called a non-singular matrix.

The laws also apply to vectors since they are $n \times 1$ (or $1 \times n$) matrices.

Identity Law:

$$IA = AI = A$$

Symmetry Law:

$$A + B = B + A$$

Associative Laws:

$$A \pm (B \pm C) = (A \pm B) \pm C$$

$$A(BC) = (AB)C$$

Distributive Laws:

$$A(B \pm C) = AB \pm AC$$

$$(A \pm B)C = AC \pm BC$$

Transposition Laws:

$$I^T = I$$

$$(A^T)^T = A$$

$$(A \pm B)^T = A^T \pm B^T$$

$$(AB)^T = B^T A^T$$

Scaling Laws:

$$kA = Ak$$

$$k(AB) = (kA)B = A(kB)$$

$$kA^T = (kA)^T$$