HOUSEHOLDER REDUCTION OF LINEAR EQUATIONS

This paper discusses Householder reduction of n linear equations to a triangular form which can be solved by back substitution. The main strength of the method is its unconditional numerical stability. I explain how Householder reduction can be derived from elementary matrix algebra. The method is illustrated by a numerical example and a Pascal procedure. I assume that you have a general knowledge of vector and matrix algebra, but are less familiar with linear transformation of a vector space.

1 INTRODUCTION

The solution of *linear equations* is important in many areas of science and engineering (Kreyszig 1988). This chapter discusses *Householder reduction* of n linear equations to a triangular form that can be solved by back substitution (Householder 1958, Press 1989). I will explain how Householder reduction can be derived from elementary matrix algebra. The method is illustrated by a numerical example and a Pascal procedure.

I assume that you have a general knowledge of vector and matrix algebra, but are less familiar with linear transformation of a vector space.

I begin by looking at Gaussian elimination.

2 GAUSSIAN ELIMINATION

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

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

P. Brinch Hansen, Householder reduction of linear equations, *ACM Computing Surveys 24*, 2 (June 1992), 185–194.

First, you eliminate x_1 from the second equation by subtracting 1/2 of the first equation from the second one. Then you eliminate x_1 from the third equation by subtracting 3/2 of the first equation from the third one. Now, you 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, you eliminate x_2 from the third equation by adding the second equation to the third one. The equations have been reduced now to a triangular form that 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, you immediately have $x_3 = 3$. By substituting this value in the second equation, you find $x_2 = 2$. Substituting these two values in the first equation you obtain $x_1 = 1$.

In general, you have n linear equations with n unknowns:

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

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

The equation system (1) can be expressed as a vector equation

$$Ax = b \tag{2}$$

where A is the $n \times n$ matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

while x and b are n-dimensional column vectors

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

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

Gaussian elimination reduces (2) to an equivalent form

$$Ux = c$$

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

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

with all zeros below the main diagonal. The elimination process replaces the original right-hand side b by another n-dimensional column vector c.

The scaling of equations is a source of numerical errors in Gaussian elimination. To eliminate the first unknown x_1 from, say, the second equation, you subtract the first equation multiplied by a_{21}/a_{11} from the second equation. However, if the *pivot element* a_{11} is very small, the scaling factor a_{21}/a_{11} becomes very large, and you 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 a process called *pivoting*: By changing the order in which the equations are written, you can make the pivot element as large as possible. You examine the first coefficient of every equation, that is

$$a_{11}, a_{21}, \cdots, a_{n1}$$

If the largest of these coefficients is, say, a_{51}, then you exchange equations 1 and 5. After this rearrangement, you subtract multiples of the (new) first equation from the remaining ones. The pivoting process is repeated for each submatrix during the Gaussian elimination.

Pivoting rearranges both the rows of the matrix and the elements of the right-hand side. The algorithm must keep track of this permutation in an additional vector. Although pivoting does not guarantee numerical stability, numerical analysts believe that it works in practice (Golub 1989, Press 1989).

In the following, I describe an alternative method that is numerically stable and does not require pivoting. This method has been used in a parallel algorithm (Brinch Hansen 1990a, 1990b).

3 SCALAR PRODUCTS

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 I will take for granted.

Let a and b be two n-dimensional column vectors:

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

The *transpose* of a and b are the row vectors

$$a^T = [\ a_1\ a_2\ \cdots\ a_n\ ]$$

$$b^T = [\ b_1\ b_2\ \cdots\ b_n\ ]$$

The *scalar product* of a and b is

$$a^T b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{3}$$

A scalar product is obviously symmetric:

$$a^T b = b^T a \tag{4}$$

The Euclidean *norm*

$$\|a\| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2} \tag{5}$$

is the length of the n-dimensional vector a.

From (3) and (5) you obtain an equivalent definition of the norm:

$$\|a\|^2 = a^T a \tag{6}$$

4 REFLECTION

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.

First, I will look at reflection in three-dimensional space. The reflection plane P includes the origin O and is perpendicular to a given vector v. For an arbitrary vector a, I wish to find another vector b, which is the reflection of a on the other side of the plane P. Figure 1 shows a plane that includes the vectors v, a, and b. The dotted line represents the reflection plane P, which is perpendicular to v.

The concept of *reflection* is defined by three equations. The reflection plane P is determined by the vector v. To simplify the algebra, I assume that v is of length 1:

$$\|v\| = 1 \tag{7}$$

Reflection preserves the norm of a vector:

$$\|a\| = \|b\| \tag{8}$$

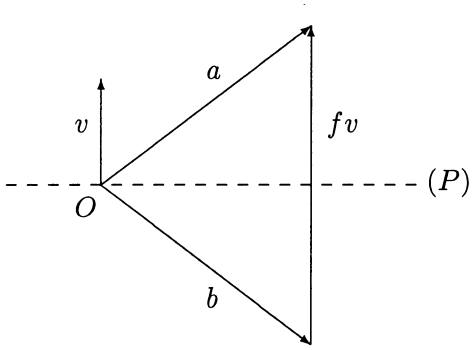

Figure 1 Reflection.

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

$$fv = a - b \tag{9}$$

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

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

$$\begin{aligned}
\|a\|^2 &= \|b\|^2 && \text{by (8)} \\
&= (a - fv)^T(a - fv) && \text{by (6), (9)} \\
&= a^Ta - fa^Tv - fv^Ta + f^2v^Tv \\
&= \|a\|^2 - 2fv^Ta + f^2 && \text{by (4), (6), (7)}
\end{aligned}$$

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

$$f = 2v^Ta \tag{10}$$

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

$$f = -2v^Tb \tag{11}$$

Finally I define b in terms of a and v

$$\begin{aligned} b &= a - vf && \text{by (9)} \\ &= Ia - v(2v^T a) && \text{by (10)} \\ &= (I - 2vv^T)a \end{aligned}$$

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

$$b = Ha \tag{12}$$

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

$$H = I - 2vv^T \tag{13}$$

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 three-dimensional space. However, the algebraic equations derived from this figure make no assumptions about the dimension of space. In the following, I will simply assume that (12) and (13) define a transformation of an n-dimensional vector. By analogy, I 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:

$$\|Ha\| = \|a\|$$

This follows from (8) and (12).

If you reflect a vector twice through the same plane, you 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 nonsingular matrix that is its own inverse:

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

(see the Appendix).

5 HOUSEHOLDER REDUCTION

I am 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,

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

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 you multiply a system of linear equations

$$Ax = b$$

by a nonsingular matrix H, you 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

$$a_1 = [\ a_{11}\ a_{21}\ \cdots\ a_{n1}\ ]^T \tag{14}$$

into a column of the form

$$Ha_1 = [\ d_{11}\ 0\ \cdots\ 0\ ]^T \tag{15}$$

where the diagonal element is

$$d_{11} = \pm\|a_1\| \tag{16}$$

The choice of sign will be made later.

Equations (14)–(16) 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

$$\begin{aligned} f_1 v &= a_1 - b_1 \qquad \text{by (9)} \\ &= a_1 - Ha_1 \quad \text{by (12)} \end{aligned}$$

Combining this with (14) and (15) you find

$$f_1 v = [\ w_{11}\ a_{21}\ \cdots\ a_{n1}\ ]^T \tag{17}$$

where the first element is

$$w_{11} = a_{11} - d_{11} \tag{18}$$

The distance between a_1 and its image Ha_1 is f_1 where

$$\begin{aligned} f_1^2 &= f_1(-2v^T H a_1) \quad \text{by (11), (12)} \\ &= -2(f_1 v)^T H a_1 \\ &= -2 w_{11} d_{11} \qquad \text{by (3), (15), (17)} \end{aligned}$$

In short,

$$f_1 = \sqrt{-2 w_{11} d_{11}} \tag{19}$$

The unit vector v that determines the appropriate Householder matrix is

$$v = f_1 v / f_1$$

or by (17):

$$v = [\ w_{11}\ a_{21}\ \cdots\ a_{n1}\ ]^T / f_1 \tag{20}$$

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 (9), (10), and (12):

$$Ha_i = a_i - f_i v \tag{21}$$

$$f_i = 2 v^T a_i \tag{22}$$

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

6 NUMERICAL STABILITY

I still need to decide which sign to use for the diagonal element d_{11} in (16).

If $d_{11} = a_{11}$, the scalars w_{11} and f_1 are zero by (18) and (19), and the division by f_1 in (20) causes overflow. You can avoid this problem by selecting the sign that makes $d_{11} \neq a_{11}$.

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

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

For $a_1 = a_{11}e_1$ there are four cases to consider:

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

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:

$$d_{11} = \text{ if } a_{11} > 0 \text{ then } -\|a_1\| \text{ else } \|a_1\| \tag{23}$$

7 COMPUTATIONAL RULES

I am now ready to summarize the rules for computing the matrix HA as defined by (3), (6), (15), and (18)–(23):

$$\begin{array}{rcl}
\|a_1\| & = & \sqrt{a_1^T a_1} \\
d_{11} & = & \text{ if } a_{11} > 0 \text{ then } -\|a_1\| \text{ else } \|a_1\| \\
w_{11} & = & a_{11} - d_{11} \\
f_1 & = & \sqrt{-2w_{11}d_{11}} \\
Ha_1 & = & [\ d_{11}\ 0\ \cdots\ 0\ ]^T \\
v & = & [\ w_{11}\ a_{21}\ \cdots\ a_{n1}\ ]^T / f_1 \\
f_i & = & 2v^T a_i \qquad \text{for } 1 < i \ \leq \ n \\
Ha_i & = & a_i - f_i v
\end{array} \tag{24}$$

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 (24), is called a *Householder transformation* (Fig. 2).

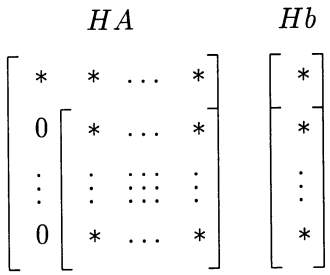

Figure 2 Householder transformation.

The second step reduces the $(n-1) \times (n-1)$ submatrix of HA, shown in Fig. 2, by Householder transformation. Now, you 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 in Fig. 2.

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.

8 A NUMERICAL EXAMPLE

I now return to the previous example of three equations with three unknowns. For convenience, I 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, you reduce $A0$ to a matrix $A1$ with all zeros below the diagonal element in the first column. This is done column by column using (24).

The numbers shown below were produced by a computer using 64-bit real arithmetic and rounded to four decimal places in the printing.

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:

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

Third column:

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

Fourth column:

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

You 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 2×3 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

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

9 PASCAL PROCEDURE

The following Pascal procedure assumes that the matrix A is stored by columns, that is, $a[i]$ denotes the ith 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;
    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 { reduce };
```

For $n \gg 1$, the *execution time* of the algorithm is dominated by the *transform* procedure, which uses one addition, one subtraction, and two multiplications per array element. The ith submatrix requires $n - i + 1$ *transform* operations, each involving $4(n - i + 1)$ arithmetic operations. So the total number of numerical operations is approximately

$$\sum_{i=1}^{n-1} 4(n-i+1)^2 = \sum_{k=2}^{n} 4k^2 \approx 4n^3/3$$

A similar analysis shows that Gaussian elimination requires $2n^3/3$ arithmetic operations only.

10 FINAL REMARKS

I have explained Householder's method for reducing a matrix to triangular form. The main advantage of the method is its unconditional stability. I have illustrated the computation by a numerical example and a Pascal procedure.

Gaussian elimination and Householder reduction of an $n \times n$ matrix both have $O(n^3)$ complexity. However, Householder reduction requires twice as many numerical operations. For that reason, Householder reduction is seldom used to solve linear equations on a sequential computer.

Why then should you be interested in Householder reduction?

1. For some matrices, Gaussian elimination with pivoting is highly inaccurate. Numerical analysts believe that such matrices are so rare that pivoting is stable "in practice." However, I have not found a theoretical or statistical justification of this claim in the literature. Householder's method is unconditionally stable, both in theory and in practice. An engineer usually prefers a stable method with reasonable speed to a faster, but potentially unstable, technique.

2. When a *multicomputer* with p processors solves n linear equations in parallel, the solution time has the form

$$T_p = an^3/p + bn^2$$

where a and b are system-dependent constants of matrix transformation and processor communication. The transformation time is reduced by the number of processors. The communication time is proportional to the number of matrix elements. Parallelism reduces the transformation time, but not the communication time. Since Gaussian elimination and Householder reduction require the same amount of communication, a multicomputer reduces the time difference between these methods. On a Computing Surface with 45 transputers, I used both methods to solve 1000 equations. The parallel solution times differed by only 50% (Brinch Hansen 1990b, 1992). For parallel solution of linear equations, Householder reduction is an attractive compromise between unconditional numerical stability and computing speed.

3. Finally, it should be mentioned that Householder reduction is used for *least squares* and *eigenvalue* computations in the *Linpack* procedures developed at Argonne National Laboratory (Dongarra 1979).

Householder reduction is an interesting example of a fundamental computation with a subtle theory and a short algorithm. If you are interested in further details and alternative methods, you will find them in the books by Golub (1989) and Press (1989).

11 APPENDIX: MATRIX ALGEBRA

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

The *identity matrix* is

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

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 *nonsingular* matrix.

The laws apply also 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:

$$\begin{aligned} A \pm (B \pm C) &= (A \pm B) \pm C \\ A(BC) &= (AB)C \end{aligned}$$

Distributive Laws:

$$\begin{aligned} A(B \pm C) &= AB \pm AC \\ (A \pm B)C &= AC \pm BC \end{aligned}$$

Transposition Laws:

$$\begin{aligned} I^T &= I \\ (A^T)^T &= A \\ (A \pm B)^T &= A^T \pm B^T \\ (AB)^T &= B^T A^T \end{aligned}$$

Scaling Laws:

$$kA = Ak$$

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

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

Acknowledgements

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

References

BRINCH HANSEN, P. 1990a. The all-pairs pipeline. School of Computer and Information Science, Syracuse University, Syracuse, N.Y.

BRINCH HANSEN, P. l990b. Balancing a pipeline by folding. School of Computer and Information Science, Syracuse University, Syracuse, N.Y.

BRINCH HANSEN, P. 1992. Unpublished measurements of Gaussian elinination on the all-pairs, folded pipeline. School of Computer and Information Science, Syracuse University, Syracuse, N.Y.

DONGARRA, J. J., BUNCH, J. R., MOLER, C. B., AND STEWART, G. W. 1979. *Linpack Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, Penn.

GOLUB, G. H., AND VAN LOAN, C. F. 1989. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryl.

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

KREYSZIG, E. 1988. *Advanced Engineering Mathematics*. John Wiley & Sons, N.Y.

PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., AND VETTERLING, W. T. 1989. *Numerical Recipes in Pascal—The Art of Scientific Computing*. Cambridge University Prese, N.Y.