Numerische Mathematik 7, 269–276 (1965)

# *Handbook Series Linear Algebra*

## Linear Least Squares Solutions by Householder Transformations***

Contributed by
PETER BUSINGER and GENE H. GOLUB

### 1. Theoretical Background

Let $A$ be a given $m \times n$ real matrix with $m \geqq n$ and of rank $n$ and $\boldsymbol{b}$ a given vector. We wish to determine a vector $\hat{\boldsymbol{x}}$ such that

$$\|\boldsymbol{b} - A\hat{\boldsymbol{x}}\| = \min.$$

where $\|\ldots\|$ indicates the euclidean norm. Since the euclidean norm is unitarily invariant

$$\|\boldsymbol{b} - A\boldsymbol{x}\| = \|\boldsymbol{c} - QA\boldsymbol{x}\|$$

where $\boldsymbol{c} = Q\boldsymbol{b}$ and $Q^T Q = I$. We choose $Q$ so that

$$QA = R = \begin{pmatrix} \tilde{R} \\ \ldots \\ 0 \end{pmatrix}_{\}(m-n)\times n} \tag{1}$$

and $\tilde{R}$ is an upper triangular matrix. Clearly,

$$\hat{\boldsymbol{x}} = \tilde{R}^{-1}\tilde{\boldsymbol{c}}$$

where $\tilde{\boldsymbol{c}}$ denotes the first $n$ components of $\boldsymbol{c}$.

A very effective method to realize the decomposition (1) is via Householder transformations [*1*]. Let $A = A^{(1)}$, and let $A^{(2)}, A^{(3)}, \ldots, A^{(n+1)}$ be defined as follows:

$$A^{(k+1)} = P^{(k)} A^{(k)} \qquad (k = 1, 2, \ldots, n).$$

$P^{(k)}$ is a symmetric, orthogonal matrix of the form

$$P^{(k)} = I - \beta_k \boldsymbol{u}^{(k)} \boldsymbol{u}^{(k)T}$$

 This report was supported in part by Office of Naval Research Contract Nonr-225(37) (NR 044-11) at Stanford University.

** *Editor's note.* In this fascicle, prepublication of algorithms from the Linear Algebra series of the Handbook for Automatic Computation is continued. Algorithms are published in ALGOL 60 reference language as approved by the IFIP. Contributions in this series should be styled after the most recently published ones. Inquiries are to be directed to the editor. F. L. BAUER, München

where the elements of $P^{(k)}$ are derived so that

$$a_{i,k}^{(k+1)}=0$$

for $i=k+1, \ldots, m$. It can be shown, cf. [2], that $P^{(k)}$ is generated as follows:

$$\sigma_k=\Big(\sum_{i=k}^{m}(a_{i,k}^{(k)})^2\Big)^{\frac{1}{2}},$$

$$\beta_k=[\sigma_k(\sigma_k+|a_{k,k}^{(k)}|)]^{-1},$$

$$u_i^{(k)}=0 \quad \text{for} \quad i<k,$$

$$u_k^{(k)}=\operatorname{sgn}(a_{k,k}^{(k)})(\sigma_k+|a_{k,k}^{(k)}|),$$

$$u_i^{(k)}=a_{i,k}^{(k)} \quad \text{for} \quad i>k.$$

The matrix $P^{(k)}$ is not computed explicitly. Rather we note that

$$A^{(k+1)}=(I-\beta_k \boldsymbol{u}^{(k)}\boldsymbol{u}^{(k)^T})A^{(k)}$$
$$=A^{(k)}-\boldsymbol{u}^{(k)}\boldsymbol{y}_k^T$$

where

$$\boldsymbol{y}_k^T=\beta_k\,\boldsymbol{u}^{(k)^T}A^{(k)}.$$

In computing the vector $\boldsymbol{y}_k$ and $A^{(k+1)}$, one takes advantage of the fact that the first $(k-1)$ components of $\boldsymbol{u}^{(k)}$ are equal to zero.

At the $k^{\text{th}}$ stage the column of $A^{(k)}$ is chosen which will maximize $|a_{k,k}^{(k+1)}|$. Let

$$s_j^{(k)}=\sum_{i=k}^{m}(a_{i,k}^{(k)})^2 \qquad j=k, k+1, \ldots, n.$$

Then since $|a_{k,k}^{(k+1)}|=\sigma_k$, one should choose that column for which $s_j^{(k)}$ is maximized. After $A^{(k+1)}$ has been computed, one can compute $s_j^{(k+1)}$ as follows:

$$s_j^{(k+1)}=s_j^{(k)}-(a_{k,j}^{(k+1)})^2$$

since the orthogonal transformations leave the column lengths invariant.

Let $\bar{\boldsymbol{x}}$ be the initial solution obtained, and let $\hat{\boldsymbol{x}}=\bar{\boldsymbol{x}}+\boldsymbol{e}$. Then

$$\|\boldsymbol{b}-A\hat{\boldsymbol{x}}\|=\|\boldsymbol{r}-A\boldsymbol{e}\|$$

where

$$\boldsymbol{r}=\boldsymbol{b}-A\bar{\boldsymbol{x}}, \quad \text{the residual vector.}$$

Thus the correction vector $\boldsymbol{e}$ is itself the solution to a linear least squares problem. Once $A$ has been decomposed, and if the transformations have been saved, then it is a simple matter to compute $\boldsymbol{r}$ and solve for $\boldsymbol{e}$. The iteration process is continued until convergence. There is no assurance, however, that all digits of the final solution will be correct.

## 2. Applicability

The algorithm *least squares solution* may be used for solving linear least squares problems, systems of linear equations where $A$ is a square matrix, and thus also for inverting matrices.

### 3. Formal Parameter List

The matrix $A$ is a given matrix of an overdetermined system of $m$ equations in $n$ unknowns. The matrix $B$ contains $p$ right hand sides, and the solution is stored in $X$. If no solution can be found then the problem is left unsolved and the emergency exit *singular* is used. The termination procedure is dependent upon *eta* which is the largest positive number such that $fl(1+\eta)\equiv 1$ where $fl(\ldots)$ indicates the floating point operation.

### 4. Algol Program

```
procedure least squares solution (a, x, b, m, n, p, eta, singular);
    value m, n, p, eta;
    array a, x, b; integer m, n, p; real eta; label singular;
comment The array a[1:m, 1:n] contains the given matrix of an overdetermined
        system of m linear equations in n unknowns (m≧n). For the p right
        hand sides given as the columns of the array b[1:m, 1:p], the least
        squares solutions are computed and stored as the columns of the
        array x[1:n, 1:p]. If rank(a)<n then the problem is left unsolved
        and the emergency exit singular is used. In either case a and b
        are left intact. Eta is the relative machine precision;
begin
    real procedure inner product (i, m, n, a, b, c);
        value m, n, c;
        real a, b, c; integer i, m, n;
    comment The body of this inner product routine should preferably be
            replaced by its double precision equivalent in machine code;
    begin
        for i:= m step 1 until n do c:= c+a×b;
        inner product:= c
    end inner product;
    procedure decompose(m, n, qr, alpha, pivot, singular);
        value m, n;
        integer m, n; array qr, alpha; integer array pivot;
        label singular; comment nonlocal real procedure inner product;
comment Decompose reduces the matrix given in the array qr[1:m, 1:n]
        (m≧n) to upper right triangular form by means of n elementary
        orthogonal transformations (I-beta uu'). The diagonal elements of
        the reduced matrix are stored in the array alpha[1:n], the off diagonal
        elements in the upper right triangular part of qr. The nonzero com-
        ponents of the vectors u are stored on and below the leading diagonal
        of qr. Pivoting is done by choosing at each step the column with
        the largest sum of squares to be reduced next. These interchanges
        are recorded in the array pivot[1:n]. If at any stage the sum of
        squares of the column to be reduced is exactly equal to zero then
        the emergency exit singular is used;
```

```
begin
    integer i, j, jbar, k; real beta, sigma, alphak, qrkk;
    array y, sum[1:n];
    for j := 1 step 1 until n do
    begin comment j-th column sum;
        sum[j] := inner product(i, 1, m, qr[i,j], qr[i,j], 0);
        pivot[j] := j
    end j-th column sum;
    for k := 1 step 1 until n do
    begin comment k-th Householder transformation;
        sigma := sum[k]; jbar := k;
        for j := k+1 step 1 until n do
            if sigma<sum[j] then
            begin
                sigma := sum[j]; jbar := j
            end;
        if jbar ≠ k then
        begin comment column interchange;
            i := pivot[k]; pivot[k] := pivot[jbar]; pivot[jbar] := i;
            sum[jbar] := sum[k]; sum[k] := sigma;
            for i := 1 step 1 until m do
            begin
                sigma := qr[i,k]; qr[i,k] := qr[i,jbar];
                qr[i,jbar] := sigma
            end i
        end column interchange;
        sigma := inner product(i, k, m, qr[i,k], qr[i,k], 0);
        if sigma = 0 then go to singular;
        qrkk := qr[k,k];
        alphak := alpha[k] := if qrkk<0 then sqrt(sigma) else -sqrt(sigma);
        beta := 1/(sigma - qrkk×alphak);
        qr[k,k] := qrkk - alphak;
        for j := k+1 step 1 until n do
            y[j] := beta×inner product(i, k, m, qr[i,k], qr[i,j], 0);
        for j := k+1 step 1 until n do
        begin
            for i := k step 1 until m do
            qr[i,j] := qr[i,j] - qr[i,k]×y[j];
            sum[j] := sum[j] - qr[k,j] ↑2
        end j
    end k-th Householder transformation
end decompose;

procedure solve(m, n, qr, alpha, pivot, r, y);
    value m, n;
    integer m, n; array qr, alpha, r, y; integer array pivot;
    comment nonlocal real procedure inner product;
```

```
comment Using the vectors u whose nonzero components are stored on and
        below the main diagonal of qr[1:m, 1:n] solve applies the n trans-
        formations (I-beta uu') to the right hand side r[1:m]. From the
        reduced matrix given in alpha[1:n] and the upper right triangular
        part of qr, solve then computes by backsubstitution an approximate
        solution to the linear system. The components of the solution vector
        are stored in y[1:n] in the order prescribed by pivot[1:n];
begin
    integer i,j; real gamma; array z[1:n];
    for j:=1 step 1 until n do
    begin comment apply the j-th transformation to the right hand side;
        gamma:=inner product(i,j,m,qr[i,j],r[i],0)/(alpha[j]×qr[j,j]);
        for i:=j step 1 until m do r[i]:=r[i]+gamma×qr[i,j]
    end j-th transformation;
    z[n]:=r[n]/alpha[n];
    for i:=n-1 step -1 until 1 do
        z[i]:=-inner product(j,i+1,n,qr[i,j],z[j],-r[i])/alpha[i];
    for i:=1 step 1 until n do y[pivot[i]]:=z[i]
end solve;
integer i, j, k; real normy0, norme0, norme1, eta2;
array qr[1:m, 1:n], alpha, e, y[1:n], r[1:m]; integer array pivot[1:n];
for j:=1 step 1 until n do
  for i:=1 step 1 until m do qr[i,j]:=a[i,j];
decompose(m, n, qr, alpha, pivot, singular);
eta2:=eta ↑2;
for k:=1 step 1 until p do
begin comment solution for the k-th right hand side;
    for i:=1 step 1 until m do r[i]:=b[i,k];
    solve(m, n, qr, alpha, pivot, r, y);
    for i:=1 step 1 until m do
        r[i]:=-inner product(j,1,n,a[i,j],y[j],-b[i,k]);
    solve(m, n, qr, alpha, pivot, r, e);
    normy0:=norme1:=0;
    for i:=1 step 1 until n do
    begin
        normy0:=normy0+y[i] ↑2; norme1:=norme1+e[i] ↑2
    end i;
    if norme1>0.0625×normy0 then go to singular;
      comment No attempt at obtaining the solution is made unless the
              norm of the first correction is significantly smaller than the
              norm of the initial solution;
improve:
    for i:=1 step 1 until n do y[i]:=y[i]+e[i];
    if norme1<eta2×normy0 then go to store;
      comment Terminate the iteration if the correction was of little signi-
              ficance;
```

```
    for i := 1 step 1 until m do
        r[i] := -inner product(j, 1, n, a[i, j], y[j], -b[i, k]);
    solve(m, n, qr, alpha, pivot, r, e);
    norme0 := norme1; norme1 := 0;
    for i := 1 step 1 until n do norme1 := norme1 + e[i] ↑2;
    if norme1 ≦ 0.0625 × norme0 then go to improve;
      comment Terminate the iteration also if the norm of the correction
              failed to decrease sufficiently as compared with the norm
              of the previous correction;
store:
    for i := 1 step 1 until n do x[i, k] := y[i]
  end k-th right hand side
end least squares solution
```

## 5. Organizational and Notational Details

The array $a$, containing the original matrix $A$, is transferred to the array $qr$ which serves as storage for $A^{(k)}$. The non-zero components of the vectors $\boldsymbol{u}^{(k)}$ are stored on and below the leading diagonal of $qr$. The diagonal elements of $\tilde{R}$, the reduced matrix, are stored in the array $\alpha$.

The column sum of squares, $s_j^{(k)}$, is stored in the array *sum*. Naturally, the elements of this array are interchanged whenever the columns of $A^{(k+1)}$ are interchanged. The array pivot contains the order in which the columns are selected.

## 6. Discussion of Numerical Properties

The program uses the iteration scheme described in section 1. Let

$$\boldsymbol{x}^{(q+1)} = \boldsymbol{x}^{(q)} + \boldsymbol{e}^{(q)}, \qquad q = 0, 1, \ldots$$

with $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ and

$$\boldsymbol{r}^{(q)} = \boldsymbol{b} - A\boldsymbol{x}^{(q)}.$$

The residual vector $\boldsymbol{r}^{(q)}$ should be computed using double precision inner products and then rounded to single precision accuracy.

If $\|\boldsymbol{e}^{(1)}\|/\|\boldsymbol{x}^{(1)}\| > 0.25$ or if $\sigma_k = 0$ at any stage then the singular exit is used. The iteration process is terminated if $\|\boldsymbol{e}^{(k+1)}\| > 0.25\|\boldsymbol{e}^{(k)}\|$ or if $\|\boldsymbol{e}^{(k)}\|/\|\boldsymbol{x}^{(0)}\| < \eta$.

## 7. Examples of the Use of the Procedure

In many statistical applications, it is necessary to compute $(A^TA)^{-1}$ or $\det(A^TA)$.

Since

$$(A^TA) = \tilde{R}^T\tilde{R}, \qquad (A^TA)^{-1} = \tilde{R}^{-1}\tilde{R}^{-T}.$$

In addition,

$$\det(A^TA) = (\det \tilde{R})^2 = r_{11}^2 \cdot r_{22}^2 \cdots r_{nn}^2.$$

## 8. Test Results

The procedure was tested on the B 5000 (Stanford University) which has a 39 bit mantissa. All inner products were computed using a double precision inner product routine. We give two examples. The first example consists of

Example I

| A | | | | | B |
|---|---|---|---|---|---|
| $3.6000000000_{10}+01$ | $-6.3000000000_{10}+02$ | $3.3600000000_{10}+03$ | $-7.5600000000_{10}+03$ | $7.5600000000_{10}+03$ | $4.6300000000_{10}+02$ |
| $-6.3000000000_{10}+02$ | $1.4700000000_{10}+04$ | $-8.8200000000_{10}+04$ | $2.1168000000_{10}+05$ | $-2.2050000000_{10}+05$ | $-1.3860000000_{10}+04$ |
| $3.3600000000_{10}+03$ | $-8.8200000000_{10}+04$ | $5.6448000000_{10}+05$ | $-1.4112000000_{10}+06$ | $1.5120000000_{10}+06$ | $9.7020000000_{10}+04$ |
| $-7.5600000000_{10}+03$ | $2.1168000000_{10}+05$ | $-1.4112000000_{10}+06$ | $3.6288000000_{10}+06$ | $-3.9690000000_{10}+06$ | $-2.5872000000_{10}+05$ |
| $7.5600000000_{10}+03$ | $-2.2050000000_{10}+05$ | $1.5120000000_{10}+06$ | $-3.9690000000_{10}+06$ | $4.4100000000_{10}+06$ | $2.9106000000_{10}+05$ |
| $-2.7720000000_{10}+03$ | $8.3160000000_{10}+04$ | $-5.8212000000_{10}+05$ | $1.5523200000_{10}+06$ | $-1.7463600000_{10}+06$ | $-1.1642400000_{10}+05$ |

$x =$ $1.0000000000_{10}+00$ $5.0000000000_{10}-01$ $3.3333333330_{10}-01$ $2.5000000000_{10}-01$ $2.0000000000_{10}-01$

Example II

| B |
|---|
| $-4.1570000000_{10}+03$ |
| $-1.7820000000_{10}+04$ |
| $9.3555000000_{10}+04$ |
| $-2.6180000000_{10}+05$ |
| $2.8828800000_{10}+05$ |
| $-1.1894400000_{10}+05$ |

$x =$ $1.0013522567_{10}+00$ $5.0045029804_{10}-01$ $3.3352602836_{10}-01$ $2.5008421208_{10}-01$ $2.0002992051_{10}-01$

the first five columns of the inverse of the $6 \times 6$ Hilbert matrix. The vector $\boldsymbol{b}$ was chosen so that $\|\boldsymbol{b} - A\boldsymbol{x}\| = 0$. In the second example, a vector orthogonal to the columns of $A$ was added to $\boldsymbol{b}$. Thus the solution should be precisely the same as in the first example. Three iterations were required in each case. Note all digits are correct of the computed solution for example I. In example II, however, only the first three digits are correct. Thus, in general not all solutions will be accurate to full working accuracy.

A second test was performed on a Telefunken TR 4 computer at the Kommission für Elektronisches Rechnen der Bayerischen Akademie der Wissenschaften, München.

## References

[1] HOUSEHOLDER, A. S.: Unitary Triangularization of a Nonsymmetric Matrix. J. Assoc. Comput. Mach. **5**, 339–342 (1958).

[2] WILKINSON, J. H.: Householder's Method for Symmetric Matrices. Numer. Math. **4**, 354–361 (1962).

Computation Center
University of Texas
and
Computer Science Division
Stanford University
Stanford, California 94305 (USA)