Algorithms for the QR-Decomposition

WALTER GANDER

RESEARCH REPORT NO. 80-02

APRIL 1980

SEMINAR FUER ANGEWANDTE MATHEMATIK
EIDGENOESSISCHE TECHNISCHE HOCHSCHULE
CH-8092 ZUERICH

\[1\text{retyped in } \LaTeX \text{ October 2003}\]
IN MEMORY OF

PROF. H. RUTISHAUSER †1970
Abstract

In this report we review the algorithms for the QR decomposition that are based on the Schmidt orthonormalization process and show how an accurate decomposition can be obtained using modified Gram Schmidt and reorthogonalization. We also show that the modified Gram Schmidt algorithm may be derived using the representation of the matrix product as a sum of matrices of rank one.

1 Introduction

Let $A$ be a real $m \times n$ matrix ($m > n$) with rank($A$) = $n$. It is well known that $A$ may be decomposed into the product

$$A = QR$$

(1)

where $Q$ is $(m \times n)$ orthogonal ($Q^T Q = I_n$) and $R$ is $(n \times n)$ upper triangular. The earliest proposal to compute this decomposition probably was to use the Schmidt orthonormalization process. It was soon observed [8] however that this algorithm is unstable and indeed, as it performs in Example 1 it must be considered an algorithm of parallelization rather than orthogonalization! In fact even the method, although we don’t recommend it, of computing $Q$ via the Cholesky decomposition of $A^T A$,$A^T A = R^T R$

and to put

$$Q = AR^{-1}$$

seems to be superior than classical Schmidt.

The “modified Gram Schmidt” algorithm was a first attempt to stabilize Schmidt’s algorithm. However, although the computed $R$ is remarkably accurate, $Q$ need not to be orthogonal at all. Nevertheless, as was pointed out by Björck [1], modified Gram Schmidt may be used to solve least squares problems.

The remedy for really making $Q$ orthogonal, as proposed by Rutishauser [5,6] is reorthogonalization. Unfortunately this method has not become very popular because at about the same time a new way to compute the decomposition 1 was proposed [4] using elementary orthogonal Householder-matrices. This algorithm is stable and the only objection is that it does not yield an explicit representation of $Q$. Instead one has all information stored to compute $Q$ or $Q^T$ acting on a given vector as an operator. When a vector is reorthogonalized the matrix $R$ should be updated. Rutishauser did not do so on the grounds that the corrections are small. Nevertheless, by including these corrections we obtain forming the product $QR$ a matrix which is numerically closer to $A$. In this paper we show how to compute these corrections. This results in a small change in Rutishauser’s algorithm which is so inexpensive that it would be a pity to omit it. In addition we derive the modified Gram Schmidt algorithm using the representation of the matrix product as sum of matrices of rank one.
2 Classical Schmidt Algorithm

We denote in this paper the $k$–th column of a matrix $A$ by $cA_k$. Similarly we shall use $rA_j$ to refer to the $j$–th row of $A$. Furthermore we shall use $\| \cdot \|$ for the euclidian norm of a vector.

The classical Schmidt algorithm consists of $n$ steps. In each of them one $cA_k$ is used to produce a new column vector of $Q$. Consider the $k$–th step. We already have $k-1$ orthonormal vectors $cQ_1, \ldots, cQ_{k-1}$.

We take $cA_k$ and compute the projections $r_{ik}cQ_i$ with $r_{ik} := cQ_i^TcA_k$.

Then we form the vector

$$b_k = cA_k - \sum_{i=1}^{k-1} r_{ik}cQ_i$$

which is orthogonal to $cQ_1, \ldots, cQ_{k-1}$ and thus we get

$$cQ_k := b_k/\|b_k\|.$$ (3)

This leads to the following algorithm where $A$ is overwritten by $Q$:

```
for k := 1 to n do
begin
    for i := 1 to k - 1 do
    begin
        s := 0
        for j := 1 to m do s := s + a_{j,i} * a_{j,k};
        r_{ik} := s;
    end;

    for i := 1 to k - 1 do
    begin
    end;

    for j := 1 to m do a_{jk} := a_{j,k} - a_{j,i} * r_{i,k};
end;

s := 0;
for j := 1 to m do s := s + a_{j,k}^2;

r_{kk} := sqrt(s);
for j := 1 to m do a_{jk} := a_{j,k}/r_{k,k};
end k
```

If we look at Example 1 we see that the resulting matrix $Q$ is not orthogonal at all. The algorithm is unstable. However the product $QR$ equals $A$ perfectly. This is not a special virtue at all since for any arbitrary given non-singular matrix $R$ we can easily compute a matrix $Q$ row by row by forward substitution from

$$R^TQ^T = A^T.$$ (4)
Of course we then have $A = QR$ but in general $Q^TQ \neq I$.

Notice that if we eliminate the three lines

```plaintext
end;
for i := 1 to k - 1 do
begin
```

of the above algorithm then we get a variant of the modified Gram Schmidt algorithm given by Schwarz and Rutishauser [7].

Assuming the existence of the $QR$ decomposition of $A$ one can derive the classical Schmidt algorithm by letting [8]:

$$A = [cA_1, \ldots, cA_n] = [cQ_1, \ldots, cQ_n] \cdot R$$

i.e.

$$cA_k = cQ_1r_{1k} + cQ_2r_{2k} + \ldots + cQ_{k-1}r_{k-1,k} + cQ_kr_{kk} \quad (6)$$

for $k := 1, \ldots, n$.

If we suppose that $Q$ and $R$ are computed column by column, the unknown in (6) are $r_{1k}, \ldots, r_{kk}$ and $cQ_k$. By multiplying (6) from the left with $cQ_i^T$ we get first

$$cQ_i^TcA_k = r_{ik}, \quad i = 1, \ldots, k - 1,$$

and $r_{kk}, cQ_k$ are obtained as in (2), (3).

3 Modified Gram Schmidt

Our aim again is to compute $Q$ and $R$ from the relation $A = QR$. But now in place of (6) we represent the matrix product as

$$A = \sum_{i=1}^{n} cQ_i rR_i^T, \quad (7)$$

i.e. as the sum of matrices of rank one. Observe that the first $i - 1$ columns of the matrix

$$cQ_i rR_i^T = \begin{pmatrix} 0 \\ \hline \end{pmatrix} \quad (8)$$

are zero because $R$ is an upper triangular matrix.

We define the matrices

$$A^{(k)} = A - \sum_{i=1}^{k-1} cQ_i rR_i^T, \quad k = 1, \ldots, n + 1 \quad (9)$$

which have the same shape as (8) (first $k - 1$ columns are zero) since it follows from (7) that also

$$A^{(k)} = \sum_{i=k}^{n} cQ_i rR_i^T. \quad (10)$$
Clearly there holds the recursion
\[ A^{(1)} = A \]
\[ A^{(k+1)} = A^{(k)} - cQ_k r R_k^T, \quad k = 1, \ldots, n \] (12)
\[ A^{(n+1)} = 0. \] (13)

We wish to exploit the recursion (12) for computing \( Q \) and \( R \). If we assume that the first \( k - 1 \) columns of \( Q \) and the first \( k - 1 \) rows of \( R \) are already known then letting \( e_k = (0, \ldots, 1, \ldots, 0)^T \) we have, using (10),
\[ A^{(k)} e_k = c A^{(k)} = \sum_{i=k}^{n} cQ_i (r R_i^T e_k) = cQ_k r_{kk}. \] (14)

Therefore
\[ r_{kk} = \| c A^{(k)} \| \quad \text{and} \quad cQ_k = c A^{(k)}/r_{kk}. \] (15)

Thus we can compute from \( A^{(k)} \) the \( k \)-th column of \( Q \). To compute the \( k \)-th row of \( R \) we note that
\[ A^{(k+1)} = A^{(k)} - cQ_k r R_k^T. \] (17)

Therefore multiplying (16) from the left by \( cQ_k^T \) we get
\[ cQ_k^T A^{(k+1)} = 0^T \] because \( cQ_k \) is orthogonal to \( cQ_i \).

Using this in (17) we have
\[ 0^T = cQ_k^T A^{(k)} - r R_k^T \]
which on rearranging yields the rule to compute the \( k \)-th row of \( R \):
\[ r R_k = A^{(k)T} cQ_k. \] (18)

Notice that equation (18) means that we may compute
\[ r_{ik} = c A_i^{(k)T} cQ_k \]
rather than
\[ r_{ik} = c A_i^T cQ_k \]
which follows from \( A = QR \). Using equations (12), (15) and (18) leads to the following algorithm:

**Modified Gram Schmidt:**

for \( k := 1 \) to \( n \) do
begin
\[ s := 0; \]
for \( j := 1 \) to \( m \) do \( s := s + a_{jk}^2 \)
\[ r_{kk} := \text{sqrt}(s); \]
for \( j := 1 \) to \( m \) do 
\( q_{jk} := a_{jk}/r_{kk}; \)

for \( i := k + 1 \) to \( n \) do 
begin 
\( s := 0; \)
for \( j := 1 \) to \( m \) do 
\( s := s + a_{ji} \cdot q_{jk}; \)
\( r_{ki} := s; \)
for \( j := 1 \) to \( m \) do 
\( a_{ji} := a_{ji} - r_{ki} \cdot q_{jk}; \)
end;
end

4 Modified Gram Schmidt: Variant of Schwarz-Rutishauser

From §3 we have

\[
A^{(j)} = A - \sum_{i=1}^{j-1} c_{i} Q_{i} R_{i}^{T} = \sum_{i=j}^{n} c_{i} Q_{i} R_{i}^{T} \tag{19}
\]

Considering the \( k \)-th column of \( A^{(j)} \) \( (j < k) \)

\[
A^{(j)} e_{k} = c A^{(j)}_{k} = c A_{k} - \sum_{i=1}^{j-1} c_{i} Q_{i} r_{ik} = \sum_{i=j}^{k} c_{i} Q_{i} r_{ik}. \tag{20}
\]

We note that the last sum in (20) has the upper bound \( k \) and not \( n \), since \( R \) is triangular. Therefore we may compute \( c A_{k}^{(k)} \) as follows:

\[
c A_{k}^{(1)} = c A_{k} \tag{21}
\]

\[
c A_{k}^{(j)} = c A_{k}^{(j-1)} - c Q_{(j-1)} r_{j-1,k} \]
\( j = 2, \ldots, k. \)

This means that \( Q \) and \( R \) are computed column by column and for each new \( r_{ik} \)

\[
c A_{k}^{(i+1)} = c A_{k}^{(i)} - c Q_{i} r_{ik}
\]

is subtracted. By doing so we perform numerically the same operations as with modified Gram Schmidt. The operations and the rounding errors are the same only the temporal sequence of the operations is different. Comparing the \( k \)-th step we have the following:
modified Gram Schmidt  

variant of Schwarz Rutishauser

already computed

updated in the $k$-th step.

While in modified Gram Schmidt the whole remaining matrix $A$ is treated in the $k$-th step and $R$ is computed row by row, the variant of Schwarz Rutishauser leaves the columns $k+1$ to $n$ of $A$ untouched and $R$ is computed column by column.

Modified Gram Schmidt can be compared to Gaussian Elimination while the Schwarz Rutishauser variant is like direct triangular decomposition. Formally the variant of Schwarz Rutishauser is obtained by omitting three lines in the classical Schmidt algorithm: we do not wait until all $r_{ik}$ are computed to subtract $a_{jk}r_{ik}$ from $a_{jk}$ but rather do so immediately after each $r_{ik}$ is computed.

What has been derived here algebraically can of course be interpreted geometrically and that is how modified Gram Schmidt is usually explained [7]. Subtracting from $cA_k$ the projections onto $cQ_i$, $i = l, \ldots, k-1$ does not change the scalar product

$$cQ_k^T cA_k = cQ_k^T \left( cA_k - \sum_{i=1}^{k-1} cQ_i r_{ik} \right) = cQ_k^T cA_k^{(k)} \quad (22)$$

because $cQ_k$ is orthogonal to $cQ_i$.

We think that the motivation to subtracting those projections to improve the algorithm is not obvious at all. The only argument is that after subtracting the vectors become shorter and that therefore cancellation gets smaller. The algebraic approach given in §3 seems more naturally to lead to a new algorithm. In spite of the fact that a better decomposition is obtained by
modified Gram Schmidt (more accurate matrix $R$) and that this algorithm may be used successfully for least squares problems [1] the matrix $Q$ is in general not orthogonal at all (see Example 2). The only way to ensure orthogonality is reorthogonalization.

5 Reorthogonalization

The reason why the columns of $Q$ depart from orthogonality to almost arbitrary extent is the extensive cancellation that can take place in forming

$$b = cA_k - \sum_{i=1}^{k-1} r_{ik} cQ_i$$

(23)

(This was pointed out in [8] for a matrix of the order two). An indication that cancellation has occurred is given if

$$||b|| << ||cA_k||.$$

In particular one may state the rule of thumb that at least one decimal digit is lost by cancellation if

$$||b|| \leq \frac{1}{10} ||cA_k||.$$  

Equation (24) is the criterion used by Rutishauser to decide whether reorthogonalization is necessary. If cancellation has taken place the vector $b$ will be inaccurate and thus fails to be orthogonal to the $cQ_i$ for $i = 1, \ldots, k-1$. If $b$ is chosen in place of $cA_k$ to be orthogonalized with respect to $cQ_1, \ldots, cQ_{k-1}$, then in general cancellation will be much smaller and the new resulting vector will be orthogonal to the $cQ_i$ unless it is linearly dependent of them.

An algorithm due to Rutishauser which incorporates this reorthogonalization has been given in [5]. In the listing below we have framed the changes from the modified Gram Schmidt variant of Schwarz-Rutishauser.

```plaintext
for k := 1 to n do begin

  t := 0; tt := 0;
  for j := 1 to m do t := t + $a_{jk}^2$

  orth: for i := 1 to k - 1 do begin
      s := 0;
      for j := 1 to m do s := s + $a_{ji} * a_{jk}$;

      if tt = 0 then r_{ik} := s; Don’t change r_{ik} when reorthogonalizing
  end i;

  tt := 0;
  for j := 1 to m do tt := tt + $a_{jk}^2$; Length of $b$ squared
end
```
if \( tt < t/100 \) then 
begin \( t := tt; \) goto orth end;

\( r_{kk} := \text{sqrt}(tt); \)

for \( j := 1 \) to \( m \) do \( a_{jk} = a_{jk}/r_{kk}; \)

end \( k \)

This algorithm works well if \( A \) is not rank deficient (see Example 3). If the matrix \( A \) turns out to be numerically rank deficient, however, this algorithm does not work. An algorithm reliable even in the presence of rank deficiency of \( A \) is given by Rutishauser as a subprogram in his algorithm for simultaneous iteration (ritzit) [6]. As a further modification, the length of \( cA_k \) is not computed explicitly but by means of the Pythagorean theorem in this algorithm. If

\[
 b = cA_k - \sum_{i=1}^{k-1} r_{ik} cQ_i
\]

then

\[
 ||cA_k||^2 = ||b||^2 + \sum_{i=1}^{k-1} r_{ik}^2
\]

since \( b \) and the \( cQ_i \) are orthogonal. We list this algorithm in a form which is consistent with the notation introduced earlier:

for \( k := 1 \) to \( n \) do 
begin 
orig := \text{true} ;
repeat: 
begin 
\( t := 0; \)
begin 
for \( i := 1 \) to \( k-1 \) do 
begin 
s := 0
for \( j := 1 \) to \( m \) do 
begin 
s := s + a_{ji} * a_{jk};
end 
end 
\( s := 0 \)
for \( j := 1 \) to \( m \) do 
begin 
s := s + a_{jk}^2;
\( t := t + s; \)
end 
end 
\( s := 0 \)
for \( j := 1 \) to \( m \) do 
begin 
s := s + a_{jk}^2;
\( t := t + s; \)
if \( (s \leq t/100) \lor (t * mc = 0) \) then 
begin 
orig := \text{false} ;
if \( s * mc = 0 \) then \( s := 0 \) else goto repeat
end
end if:
\( r_{kk} := s := \text{sqrt}(s); \)
end if:
\( s \neq 0 \) then \( s := 1/s; \)
for \( j := 1 \) to \( m \) do 
begin 
s := s * a_{jk};
end 
end for \( k \)
Here $mc$ is the smallest positive number in the machine such that $1 + mc \neq 1$. If a column vector turns out to be numerically linear dependent then the corresponding column in $Q$ becomes the zero vector.

Of course the numerical rank of $A$ is not identical with the true rank of $A$. Therefore in general rounding errors will cause every matrix $A$ to have full rank. One indication for numerical rank deficiency are very small elements in the diagonal of $R$.

[We admit that e.g. for the Hilbert matrix (with full rank) it is not easy to decide what is “small”, (see Example 4)]. We should also mention a remark of Wilkinson [8] that though $Q$ now is orthogonal and $A$ is represented well by the product $QR$, it is by no means certain that the $Q$ and $R$ computed numerically are close to the exact $Q$ and $R$.

We see from Example 4 that with this algorithm we obtain a nice orthogonal matrix $Q$ and a fairly well representation of $A = QR$. We can improve the latter representation by including corrections in $R$ when reorthogonalizing.

6 Reorthogonalization with update of $R$

Assume for a moment that the matrix $B$ has $k$ columns where the first $k - 1$ columns are orthogonal. The $QR$ decomposition of $B$ is then given by

$$B = [cQ_1, \ldots, cQ_{k-1}, b] = [cQ_1, \ldots, cQ_{k-1}, u] \cdot R_1 =: QR_1$$

where

$$R_1 = \begin{bmatrix}
1 & 0 & \cdots & \cdots & 0 & d_1 \\
1 & 0 & \cdots & 0 & d_2 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & 0 & \vdots \\
1 & d_{k-1} & \cdots \\
d_k & & & & & \\
\end{bmatrix}$$

The $d_i$ are given by $d_i = b^T cQ_i$, $i = 1, \ldots, k - 1$ and

$$d_k = \left\| b - \sum_{i=1}^{k-1} d_i cQ_i \right\|.$$ 

Now suppose that $A = [cA_1, \ldots, cA_k]$ has been decomposed as $A = BR_2$ and we wish to reorthogonalize the last column of $B$, i.e. to compute $B = QR_1$ as above. Then there follows

$$A = BR_2 = QR_1 R_2 = QR \quad \text{with} \quad R = R_1 R_2.$$ 

The corrected matrix $R$ is a product of two upper triangular matrices and thus again upper triangular and since

$$R_1 = I + de_k^T$$

we can compute the product $R_1 R_2$ simply by

$$R = R_1 R_2 = (I + de_k^T) R_2 = R_2 + r_{kk}^{(2)} de_k^T.$$
Equation (31) means that we have to add to the last column of $R_2$ the vector $d$ multiplied by the factor $r_{kk}^{(2)}$. The vector $d$ contains the corrections of the reorthogonalization. A further simplification is obtained if we don’t normalize the vector $b$ before reorthogonalizing. This has the effect that $r_{kk}^{(2)}$, the last diagonal element of the matrix $R_2$, is 1 and therefore the corrections consist simply in adding $r_{ik} := r_{ik}^{(2)} + d_i$ for $i = 1, \ldots, k - 1$.

Formally the last algorithm presented in §4 has to be changed very little. Instead of the framed line

\[
\text{if orig then } r_{ik} := s;
\]

we now just have to put the line

\[
\text{if orig then } r_{ik} := s \text{ else } r_{ik} := r_{ik} + s;
\]

To add the corrections we thus require only $k - 1$ more additions in the $k$-th step. If we look at Example 5 we see that the product $QR$ now represents $A$ even better.

### 7 QR Decomposition with Householder-Transformations

To be complete and to compare the results obtained with Gram Schmidt orthogonalization we list an algorithm from [3] that computes the $QR$ decomposition using elementary orthogonal Householder matrices. Here we decompose

\[
P_nP_{n-1}\cdots P_1 A = \begin{pmatrix} R \hfill \\
0 \hfill 
\end{pmatrix} \tag{32}
\]

where

\[
P_i = I - w_i w_i^T \tag{33}
\]

with

\[
w_i = \begin{pmatrix}
0 \\
\vdots \\
0 \\
-1 & \cdots & x \\
\vdots \\
x \\
0 \\
\end{pmatrix} \quad \text{and} \quad ||w_i|| = \sqrt{2}. \tag{34}
\]

The matrix $Q$ is not given explicitly but we have

\[
Q^T = P_nP_{n-1}\cdots P_1.
\]
The following algorithm computes the vectors $w_i$ and the matrix $R$, overwriting $A$ as follows:

$$A = \begin{bmatrix} w_1 & w_2 & w_3 & \cdots & w_n \end{bmatrix}$$

We store only the nonzero part of $w_i$. The diagonal of $R$ is stored in the vector $d$.

for $j := 1$ to $n$ do
begin
  $s := 0$;
  for $i := j$ to $m$ do $s := s + a_{ij}^2$;
  $s := \sqrt{s}$; $d_j := \text{if } a_{jj} > 0 \text{ then } -s \text{ else } s$;
  fak := $\sqrt{s \ast (s + \text{abs}(a_{jj}))}$;
  $a_{jj} := a_{jj} - d_j$;
  for $k := j$ to $m$ do $a_{kj} := a_{kj}/\text{fak}$;
  for $i := j + 1$ to $n$ do
begin
  $s := 0$;
  for $k := j$ to $m$ do $s := s + a_{kj} \ast y_k$;
  for $k := j$ to $m$ do $y_k := y_k + a_{kj} \ast s$;
end
for $i$;
end for $j$;

No measures have been taken to stop the computation if a column turns out to be numerically linearly dependent (fak would become zero in this case).

After this decomposition we can compute $y := Q^T y$ using the vectors $w_i$ as follows:

for $j := 1$ to $n$ do
begin
  $s := 0$;
  for $k := j$ to $m$ do $s := s + a_{kj} \ast y_k$;
  for $k := j$ to $m$ do $y_k := y_k + a_{kj} \ast s$;
end

If we wish to compute $y := Q y$ the only change we have to make is to turn the loop for $j$ in (35):

for $j := n$ downto $1$ do

(35)
This is explained by

\[
Q^T = P_n P_{n-1} \cdots P_1 \\
Q = (P_n \cdots P_1)^T = P_1^T P_2^T \cdots P_n^T \\
= P_1 P_2 \cdots P_n \quad \text{since the } P_i \text{ are symmetric.}
\]

To obtain \( Q \) explicitly we have to use the algorithm (35) for \( y = e_k \) and thus we compute \( rQ_k = Q^T e_k \). The result is shown in Example 6. We see that we have good matching \( QR = A \) and nice orthogonality.

\section{Final remarks}

We are aware that reorthogonalization doubles the computing effort and that for this reason the method of Householder is usually preferred. The situation is not quite as bad for Gram Schmidt if we do actually need the matrix \( Q \) explicitly. Gram Schmidt has the advantage that we can make a vector as well orthogonal as we wish. We refer the reader to the process of “superorthogonalization” discussed and implemented in [2].

\section*{References}


Appendix: Numerical Examples

The following examples were all computed on a HP 9825 desk computer with 12 decimal digits mantissa (thus $mc = 0.5 \times 10^{-11}$). As matrix $A$ we chose the $15 \times 10$ Hilbert matrix i.e. $a_{ij} = 1/(i + j - 1)$. To save printing space we have listed only 5 decimal digits of the elements of $Q$ and $R$. To compare the accuracy we have also computed the matrices

i) $A - QR$

ii) $Q^T Q - I$

iii) $Q^T A - R$

iv) $AR^{-1} - Q$.

Each of them should be the zero matrix. For simplicity we look only for the absolute largest element and print it out. Except for classical Schmidt the matrix iv) is not zero at all. This is because $r_{10,10} \simeq 10^{-11}$ and the matrix $R$ is therefore numerically singular. The matrix ii) is a measure for the orthogonality and one may e.g. see that $Q$ is not orthogonal for the modified Gram Schmidt algorithm. Notice that the largest element of matrix i) is 10 times smaller if we update $R$ (see Example 4 and 5). The decomposition using Householder-transformations is perfect in this example (see Example 6). If we don’t use the criterion (24) but simply reorthogonalize each vector once we get the values of Example 7. They are the best for this example.
Example 1

Classical Schmidt matrix Q

7.9545E-01  5.5462E-01  2.2956E-01  7.9240E-02  2.4210E-02  1.3571E-02  1.3176E-02  1.2991E-02  1.2873E-02  1.2788E-02
3.9772E-01  2.1865E-01  6.2903E-01  5.3328E-01  3.0383E-01  2.2342E-01  2.1864E-01  2.1635E-01  2.1486E-01  2.1378E-01
1.3257E-01  2.6182E-01  1.2626E-01  2.2975E-01  2.7882E-01  2.7551E-01  2.7199E-01  2.7052E-01  2.6970E-01  2.6918E-01
1.1364E-01  2.3960E-01  1.7393E-01  1.1765E-01  3.1399E-01  5.5885E-02  5.0238E-02  4.7930E-02  4.6589E-02  4.5723E-02
8.8383E-02  2.0289E-01  2.1733E-01  6.1142E-02  1.0931E-01  2.1278E-01  2.1818E-01  2.2073E-01  2.2236E-01  2.2352E-01
7.9545E-02  1.8801E-01  2.2532E-01  1.2603E-01  9.1598E-02  2.3242E-01  2.3621E-01  2.3814E-01  2.3945E-01  2.4046E-01
7.2313E-02  1.7502E-01  2.2845E-01  1.7744E-01  9.7865E-03  1.9323E-01  1.9476E-01  1.9570E-01  1.9648E-01  1.9705E-01
6.6287E-02  1.6316E-01  2.2847E-01  2.1782E-01  1.0729E-01  1.1237E-01  1.1127E-01  1.1095E-01  1.1087E-01  1.1090E-01
6.1188E-02  1.5354E-01  2.2647E-01  2.4935E-01  1.9745E-01  4.4641E-03  5.2808E-04  1.2264E-03  2.2785E-03  2.9935E-03
5.6818E-02  1.4460E-01  2.2318E-01  2.7379E-01  2.7884E-01  1.1918E-01  1.2602E-01  1.2932E-01  1.3146E-01  1.3303E-01
5.3030E-02  1.3661E-01  2.1908E-01  2.9259E-01  3.5117E-01  2.5303E-01  2.6000E-01  2.6489E-01  2.6819E-01  2.7069E-01

These 3 vectors are almost parallel.

matrix R

1.2572E+00  7.4573E-01  5.4833E-01  4.3921E-01  3.6865E-01  3.1879E-01  2.8144E-01  2.5230E-01  2.2887E-01  2.0958E-01
1.6802E-01  1.9205E-01  1.9013E-01  1.8167E-01  1.7116E-01  1.6186E-01  1.5264E-01  1.4417E-01  1.3646E-01  1.3646E-01
1.3128E-03  2.9004E-03  4.3701E-03  4.7228E-03  5.6134E-03  6.6233E-03  7.4254E-03  8.0526E-03  8.3286E-03
2.0521E-04  2.2807E-04  2.7242E-04  3.0835E-04  5.0920E-04  2.6347E-04  2.9763E-04
8.0920E-04  2.6347E-04  2.9763E-04

1.0000E-12 = maximum element of \|A-Q*R\|
9.9999E-01 = maximum element of \|Q^tQ-I\|
1.6006E-03 = maximum element of \|Q^tA-R\|
2.1405E-06 = maximum element of \|A^*R^{-1}Q\|
Example 2  
modified Gram Schmidt  

<table>
<thead>
<tr>
<th>matrix Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.9545E-01-5.5462E-01 2.296E-01-7.924E-02 2.4235E-02-6.703E-03 1.6894E-03-7.0097E-04 1.4308E-02-4.4934E-01</td>
</tr>
<tr>
<td>3.9772E-01 2.1865E-01-6.2933E-01 5.3328E-01-3.0401E-01 1.3641E-01-5.1598E-02 1.654E-02-1.9602E-03 1.3017E-01</td>
</tr>
<tr>
<td>2.6515E-01 3.1109E-01 3.2049E-01 2.2711E-01 5.4756E-01 5.9311E-01 3.1411E-01 1.5162E-01 6.0334E-02 7.9158E-02</td>
</tr>
<tr>
<td>1.3257E-01 2.6182E-01 1.2626E-01-2.2975E-02 2.7874E-01 1.9329E-01 2.9674E-01 3.1667E-01 1.2401E-01 4.0187E-01</td>
</tr>
<tr>
<td>1.1364E-01 2.3960E-01 1.7393E-01-1.1765E-03 3.1395E-01-5.0092E-02 3.3551E-01 5.2848E-02-3.8354E-01 2.9518E-02</td>
</tr>
<tr>
<td>8.8383E-02 2.0289E-02 2.1713E-01 6.1142E-02-1.8983E-02 2.8982E-01 5.8831E-02 2.8208E-01 2.3396E-01-2.0082E-01</td>
</tr>
<tr>
<td>7.2313E-02 1.7502E-01 2.2845E-01 1.7744E-01 9.7198E-03-1.9936E-01 2.9691E-01 1.6140E-01 1.4324E-01 2.9177E-01</td>
</tr>
<tr>
<td>6.6287E-02 1.6361E-01 2.2847E-01 2.1782E-01 1.0721E-01-7.9874E-02 2.5660E-01 3.0332E-01 1.4934E-01 9.5534E-02</td>
</tr>
<tr>
<td>6.1188E-02 1.5354E-01 2.2647E-01 2.4935E-01 1.9736E-01 6.5633E-02-1.1542E-01 2.7703E-01-3.3792E-01 2.3124E-01</td>
</tr>
<tr>
<td>5.681E-02 1.4460E-01 2.2318E-01 2.7379E-01 2.7874E-01 2.2465E-01 1.0971E-01-5.0605E-02 2.2419E-01 3.0721E-01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>matrix R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2572E+00 7.4573E-01 5.4833E-01 4.3921E-01 3.6865E-01 3.1879E-01 2.8144E-01 2.5230E-01 2.2887E-01 2.0958E-01</td>
</tr>
<tr>
<td>1.6802E-01 1.9205E-01 1.9013E-01 1.8167E-01 1.7171E-01 1.6186E-01 1.5264E-01 1.4417E-01 1.3646E-01 1.3646E-01</td>
</tr>
<tr>
<td>1.3128E-03 2.9004E-03 4.3702E-03 5.6135E-03 6.6234E-03 7.4254E-03 8.0526E-03 8.0526E-03 8.0526E-03 8.0526E-03</td>
</tr>
<tr>
<td>3.5682E-07 1.3293E-06 2.9956E-06 5.3065E-06 5.3065E-06 5.3065E-06 5.3065E-06 5.3065E-06 5.3065E-06 5.3065E-06 5.3065E-06</td>
</tr>
<tr>
<td>2.0000E-12 = maximum element of (</td>
</tr>
<tr>
<td>4.4893E-01 = maximum element of (</td>
</tr>
<tr>
<td>5.643E-01 = maximum element of (</td>
</tr>
<tr>
<td>1.9017E-01 = maximum element of (</td>
</tr>
</tbody>
</table>
Example 3

Rutishauser's Schmidt from Description of ALGOL 60

column 3 reorthogonalized
column 4 reorthogonalized
column 5 reorthogonalized
column 6 reorthogonalized
column 7 reorthogonalized
column 8 reorthogonalized
column 9 reorthogonalized
column 10 reorthogonalized

matrix Q

| 7.9545E-01 | 5.5462E-01 | 2.2966E-01 | 7.9240E-02 | 2.4235E-02 | 6.7002E-03 | 1.6854E-03 | 3.8590E-04 | 8.0200E-05 | 1.5072E-05 |
| 3.9772E-01 | 2.1865E-01 | 6.2930E-01 | 5.3328E-01 | 3.0401E-01 | 1.3640E-01 | 5.1149E-02 | 1.6475E-02 | 4.6163E-03 | 1.1345E-03 |
| 2.6515E-01 | 3.1109E-01 | 3.2049E-01 | 2.2711E-01 | 1.5476E-01 | 5.0311E-01 | 3.1412E-01 | 1.5735E-01 | 5.0741E-02 | 1.9776E-02 |
| 1.3257E-01 | 2.6182E-01 | 1.2626E-01 | 2.2975E-01 | 1.2874E-01 | 1.9329E-01 | 2.9675E-01 | 3.1376E-01 | 1.2379E-01 | 4.8089E-01 |
| 1.1364E-01 | 2.3960E-01 | 1.7393E-01 | 1.1765E-01 | 3.1395E-01 | 5.0092E-02 | 3.3552E-01 | 8.2768E-02 | 3.8332E-01 | 5.5053E-02 |
| 8.8383E-02 | 2.0289E-01 | 2.1738E-01 | 6.1142E-01 | 2.8982E-01 | 2.3982E-01 | 5.8826E-02 | 2.8201E-01 | 2.3419E-01 | 2.1524E-01 |
| 7.9545E-02 | 1.9200E-01 | 1.7171E-01 | 6.1428E-01 | 1.9833E-01 | 1.0314E-01 | 1.9978E-02 | 1.9578E-01 | 1.9578E-01 | 1.9578E-01 |
| 7.2313E-02 | 1.7502E-01 | 2.2845E-01 | 1.7744E-01 | 9.7198E-03 | 1.9936E-01 | 2.9690E-01 | 1.6146E-01 | 1.4373E-01 | 3.3403E-01 |
| 6.6287E-02 | 1.6361E-01 | 2.2847E-01 | 2.1782E-01 | 1.0721E-01 | 7.9874E-02 | 2.5660E-01 | 3.0337E-01 | 1.4947E-01 | 1.3236E-01 |
| 6.1188E-02 | 1.5354E-01 | 2.2647E-01 | 2.4935E-01 | 1.9736E-01 | 6.5603E-02 | 1.1542E-01 | 2.7788E-01 | 3.3774E-01 | 2.5123E-01 |
| 5.6818E-02 | 1.4460E-01 | 2.2318E-01 | 2.7379E-01 | 2.7874E-01 | 2.2465E-01 | 1.0971E-01 | 5.0649E-02 | 2.2400E-01 | 3.4194E-01 |

matrix R

| 1.2572E 00 | 7.4573E-01 | 5.4833E-01 | 4.3921E-01 | 3.6865E-01 | 3.1879E-01 | 2.8144E-01 | 2.5230E-01 | 2.2887E-01 | 2.0958E-01 |
| 1.6802E 01 | 1.9205E-01 | 1.9013E-01 | 1.8167E-01 | 1.7171E-01 | 1.6186E-01 | 1.5264E-01 | 1.4417E-01 | 1.3646E-01 | 1.3646E-01 |
| 1.3128E-03 | 2.9004E-03 | 4.3702E-03 | 5.6135E-03 | 6.6234E-03 | 7.4254E-03 | 8.0526E-03 | 8.5874E-04 | 6.1418E-03 | 6.1418E-03 |

1.4000E-11 = maximum element of |A-Q*Q|
1.3561E-10 = maximum element of |Q*Qt-I|
1.6786E-10 = maximum element of |Q*Qt-R|
2.6597E-01 = maximum element of |A*Qt*(-1)-Q|
### Example 4

**Rutishauser's Schmidt from RITZ**

<table>
<thead>
<tr>
<th>Column</th>
<th>Orthogonalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>reorthogonalized</td>
</tr>
<tr>
<td>4</td>
<td>reorthogonalized</td>
</tr>
<tr>
<td>5</td>
<td>reorthogonalized</td>
</tr>
<tr>
<td>6</td>
<td>reorthogonalized</td>
</tr>
<tr>
<td>7</td>
<td>reorthogonalized</td>
</tr>
<tr>
<td>8</td>
<td>reorthogonalized</td>
</tr>
<tr>
<td>9</td>
<td>reorthogonalized</td>
</tr>
<tr>
<td>10</td>
<td>reorthogonalized</td>
</tr>
</tbody>
</table>

**Matrix Q**

\[
\begin{bmatrix}
2.6515E-01 & 3.1109E-01 & 3.2049E-01 & 2.2711E-01 & 5.4756E-01 & 5.0311E-01 & 3.1412E-01 & 1.6173E-01 & 5.9739E-02 & 1.9817E-02 \\
\end{bmatrix}

**Matrix R**

\[
\begin{bmatrix}
1.6802E+01 & 1.9205E-01 & 1.9013E-01 & 1.8167E-01 & 1.7171E-01 & 1.6186E-01 & 1.5264E-01 & 1.4417E-01 & 1.3646E-01 \\
1.3192E-03 & 2.9004E-03 & 4.3702E-03 & 5.6135E-03 & 6.6234E-03 & 7.4254E-03 & 8.0526E-03 & 8.7524E-03 & 9.0412E-03 \\
1.4000E-11 & \text{maximum element of } |A-Q^*R| & 3.5682E-07 & 1.3293E-06 & 2.9956E-06 & 5.3065E-06 & 7.7446E-08 & 1.9401E-07 & 8.4711E-10 & 3.9797E-09 \\
1.3571E-10 & \text{maximum element of } |O^*Q| & 1.8446E-08 & 7.7744E-08 & 1.9401E-07 & 8.4711E-10 & 3.9797E-09 & 3.5023E-11
\end{bmatrix}
\]
Example 5
from RITZIT with update of R

column  3  reorthogonalized

column  4  reorthogonalized

column  5  reorthogonalized

column  6  reorthogonalized

column  7  reorthogonalized

column  8  reorthogonalized

column  9  reorthogonalized

column 10  reorthogonalized

matrix Q

\[
\begin{array}{l}
7.9545E-01-5.5462E-01 2.2966E-01-7.9240E-02 2.4235E-02-6.7002E-03 1.6854E-03-3.8590E-04 8.0198E-05-1.5087E-05 \\
3.9772E-01 2.1865E-01-6.2903E-01 5.3328E-01-3.0401E-01 1.3640E-01-5.1149E-02 1.6475E-02-4.6162E-03 1.1362E-03 \\
2.6515E-01 3.1109E-01-3.2049E-01 2.2711E-01 5.4756E-01-5.0311E-01 3.1412E-01-1.5173E-01 5.9739E-01-2.9817E-02 \\
1.3257E-01 2.6182E-01 1.2626E-01-2.2975E-01 2.7874E-01 1.9329E-01 2.9675E-01-3.1375E-01 1.2373E-01 4.8389E-01 \\
1.1364E-01 2.3966E-01 1.7393E-01-1.1765E-01 3.1395E-01-5.0092E-02 3.3552E-01 8.2774E-02-3.8335E-01 5.5099E-02 \\
8.8383E-02 2.0289E-01 2.1733E-01 6.1142E-02-1.8938E-01 2.8982E-01 5.8826E-02 2.8201E-01 2.3410E-01-2.0848E-01 \\
7.2311E-02 1.7502E-01 2.2845E-01 1.7744E-01 9.7198E-03-1.9936E-01 2.9691E-01-1.6143E-01 1.4365E-01 3.1965E-01 \\
6.6287E-02 1.6361E-02 2.2847E-01 2.1782E-01 1.0721E-01-7.9874E-02-2.5660E-01-3.0338E-01-1.4927E-01 1.3177E-01 \\
6.1189E-02 1.5354E-01 2.2647E-01 2.4935E-01 1.9736E-01 6.5603E-02-2.1542E-01-2.7788E-01 3.3808E-01-2.4045E-01 \\
5.6818E-02 1.4460E-01 2.2318E-01 2.7379E-01 2.7874E-01 2.2465E-01 1.0971E-01-5.0565E-02-2.2377E-01 3.4794E-01 \\
\end{array}
\]

The printed digits don't differ from those of example 4 but A = QR is better represented.

matrix R

\[
\begin{array}{l}
1.2572E+00 7.4573E-01 5.4833E-01 4.3921E-01 3.6856E-01 3.1879E-01 2.8144E-01 2.5230E-01 2.2887E-01 2.0958E-01 \\
1.6802E-01 1.9255E-01 1.9013E-01 1.8167E-01 1.7171E-01 1.6186E-01 1.5264E-01 1.4417E-01 1.3646E-01 1.2882E-01 \\
1.3128E-03 2.9004E-03 4.1702E-03 5.6135E-03 6.6234E-03 7.4254E-03 8.0526E-03 9.5233E-05 2.6549E-04 6.1646E-04 2.5954E-04 \\
1.0000E-12 = maximum element of |A-Q*Q'|
1.3571E-10 = maximum element of |Q*Q'| \\
1.6796E-10 = maximum element of |Q*a-R|
4.5018E-01 = maximum element of |A*R'*(-1)-Q| \\
\end{array}
\]
Example 6

with Householder-transformation

matrix Q

\[
\begin{pmatrix}
\end{pmatrix}
\]

matrix R

\[
\begin{pmatrix}
-1.2572E \ 00 & -7.4573E \ -01 & -5.4833E \ -01 & -4.3921E \ -01 & -3.6865E \ -01 & -3.1879E \ -01 & -2.8144E \ -01 & -2.5230E \ -01 & -2.2887E \ -01 & -2.0958E \ -01 \\
-1.6039E \ -02 & -1.9205E \ -01 & -1.9013E \ -01 & -1.8172E \ -01 & -1.7171E \ -01 & -1.6186E \ -01 & -1.5264E \ -01 & -1.4417E \ -01 & -1.3646E \ -01 \\
1.6039E \ -02 & -2.6956E \ -02 & 3.3695E \ -02 & 3.7724E \ -02 & 4.0036E \ -02 & 4.1248E \ -02 & 4.1743E \ -02 & 4.1767E \ -02 \\
1.3128E \ -03 & -2.9004E \ -03 & 4.3702E \ -03 & 5.6135E \ -03 & 6.6234E \ -03 & 7.4254E \ -03 & 8.0526E \ -03 \\
9.5233E \ -05 & 2.5942E \ -04 & 8.5644E \ -04 & 6.6111E \ -04 & 8.5874E \ -04 & 1.0418E \ -03 \\
6.1646E \ -06 & 1.9905E \ -05 & 3.9975E \ -05 & 6.4359E \ -05 & 9.1280E \ -05 & -3.5682E \ -07 & -1.3293E \ -06 & -2.9956E \ -06 & -5.3065E \ -06 \\
6.0000E \ -11 & = maximum element of |A-Q*R| \\
7.0000E \ -11 & = maximum element of |Q^TQ-I| \\
2.0000E \ -11 & = maximum element of |Q^TA-R| \\
7.1999E \ -01 & = maximum element of |A^R^*(-1)-Q|
\end{pmatrix}
\]
Example 7

mod. Gram Schmidt with one reorthog. and update of R

matrix Q

\[ 7.9545E-01 \cdot 5.5462E-01 \cdot 2.2966E-01 \cdot 7.9240E-02 \cdot 2.4235E-02 \cdot 6.7002E-03 \cdot 1.6854E-03 \cdot 3.8590E-04 \cdot 8.0185E-05 \cdot 1.5061E-05 \\
3.9772E-01 \cdot 2.1865E-01 \cdot 6.2903E-01 \cdot 5.3328E-01 \cdot 3.0416E-01 \cdot 1.3645E-01 \cdot 5.1149E-02 \cdot 1.6475E-02 \cdot 4.6154E-03 \cdot 1.1326E-03 \\
2.6515E-01 \cdot 3.1109E-01 \cdot 3.2049E-01 \cdot 2.2711E-01 \cdot 5.4756E-01 \cdot 5.0311E-01 \cdot 3.1412E-01 \cdot 1.5173E-01 \cdot 5.9727E-02 \cdot 1.9722E-02 \\
1.9886E-01 \cdot 3.0771E-01 \cdot 9.1833E-02 \cdot 3.9256E-01 \cdot 2.2012E-01 \cdot 2.6734E-01 \cdot 5.1975E-01 \cdot 4.6166E-01 \cdot 2.8265E-01 \cdot 1.3355E-01 \\
1.5909E-01 \cdot 2.8585E-01 \cdot 4.5388E-02 \cdot 3.3962E-02 \cdot 1.1287E-01 \cdot 3.9628E-01 \cdot 8.2987E-02 \cdot 3.5655E-01 \cdot 5.0903E-01 \cdot 4.0301E-01 \\
1.3257E-01 \cdot 2.6182E-01 \cdot 1.2626E-01 \cdot 2.2975E-01 \cdot 2.7874E-01 \cdot 1.9329E-01 \cdot 2.9675E-01 \cdot 3.1376E-01 \cdot 1.2337E-01 \cdot 4.7718E-01 \\
1.1364E-01 \cdot 2.3960E-01 \cdot 1.7393E-01 \cdot 1.1765E-01 \cdot 3.1395E-01 \cdot 5.0092E-02 \cdot 3.3552E-02 \cdot 8.2780E-02 \cdot 3.8337E-02 \cdot 5.4303E-02 \\
9.9431E-02 \cdot 2.1998E-02 \cdot 2.0172E-01 \cdot 1.9661E-02 \cdot 2.7166E-01 \cdot 2.1865E-01 \cdot 1.6306E-01 \cdot 3.1843E-01 \cdot 1.0665E-01 \cdot 3.4282E-01 \\
8.8383E-02 \cdot 2.0289E-01 \cdot 2.1733E-01 \cdot 6.1142E-02 \cdot 1.8983E-02 \cdot 2.8982E-02 \cdot 5.8025E-02 \cdot 2.8200E-02 \cdot 2.3429E-02 \cdot 2.1990E-01 \\
7.9545E-02 \cdot 1.8801E-01 \cdot 2.2532E-01 \cdot 1.2603E-01 \cdot 9.1646E-02 \cdot 2.7620E-02 \cdot 2.2757E-01 \cdot 7.3419E-02 \cdot 3.2433E-01 \cdot 1.7612E-01 \\
7.2313E-02 \cdot 1.7502E-01 \cdot 2.2845E-01 \cdot 1.7744E-01 \cdot 9.7198E-03 \cdot 1.9936E-01 \cdot 2.9690E-01 \cdot 1.6146E-01 \cdot 1.4407E-01 \cdot 3.2024E-01 \\
6.6287E-02 \cdot 1.6361E-01 \cdot 2.2847E-01 \cdot 2.1782E-01 \cdot 1.0721E-01 \cdot 7.4974E-02 \cdot 2.5660E-02 \cdot 1.3035E-02 \cdot 1.5004E-01 \cdot 1.5829E-01 \\
6.1188E-02 \cdot 1.5354E-01 \cdot 2.2647E-01 \cdot 2.4935E-01 \cdot 1.9736E-01 \cdot 6.5603E-02 \cdot 1.1542E-02 \cdot 2.7787E-01 \cdot 3.3782E-02 \cdot 2.6004E-01 \\
5.6818E-02 \cdot 1.4460E-02 \cdot 2.2318E-02 \cdot 2.7379E-02 \cdot 2.7874E-02 \cdot 2.2456E-01 \cdot 1.0971E-01 \cdot 5.0651E-02 \cdot 2.2358E-01 \cdot 3.4888E-01 \\
5.3030E-02 \cdot 1.3661E-02 \cdot 2.1989E-01 \cdot 2.9259E-01 \cdot 3.5107E-01 \cdot 3.8841E-01 \cdot 3.9983E-01 \cdot 3.8336E-01 \cdot 3.4109E-01 \cdot 2.7362E-02

matrix R

\[ 1.2572E \ 00 \ \ 7.4573E-01 \ \ 5.4833E-01 \ \ 4.3921E-01 \ \ 3.6865E-01 \ \ 3.1879E-01 \ \ 2.8144E-01 \ \ 2.5230E-01 \ \ 2.2887E-01 \ \ 2.0958E-01 \\
1.6802E \ 01 \ \ 1.9235E-01 \ \ 1.9013E-01 \ \ 1.8167E-01 \ \ 1.7171E-01 \ \ 1.6186E-01 \ \ 1.5256E-01 \ \ 1.4417E-01 \ \ 1.3646E-01 \\
1.6019E-02 \ \ 2.6956E-02 \ \ 3.3695E-02 \ \ 2.7724E-02 \ \ 4.0036E-02 \ \ 4.1248E-02 \ \ 4.1743E-02 \ \ 4.1767E-02 \\
1.3128E-03 \ \ 2.3094E-03 \ \ 4.3702E-03 \ \ 5.6135E-03 \ \ 6.6234E-03 \ \ 7.4254E-03 \ \ 8.0526E-03 \\
9.5233E-05 \ \ 2.5942E-04 \ \ 4.5644E-04 \ \ 6.6111E-04 \ \ 8.5874E-04 \ \ 1.0418E-03 \\
6.1646E-06 \ \ 1.9905E-05 \ \ 3.9975E-05 \ \ 6.4389E-05 \ \ 9.1280E-05 \\
3.5682E-07 \ \ 1.3293E-06 \ \ 2.9956E-06 \ \ 5.3065E-06 \\
1.8445E-08 \ \ 7.7373E-08 \ \ 1.901E-07 \ \ 8.4720E-10 \ \ 3.9804E-09
\]

1.0000E-12 = maximum element of \( |A - Q^*R| \)
4.0000E-11 = maximum element of \( |Q - Q^*I| \)
4.0000E-11 = maximum element of \( |Q - A^R| \)
3.5645E-01 = maximum element of \( |A^R - (-1)^{-1}Q| \)
9 Appendix 2: Recomputation using Matlab

We have recomputed the experiments using Matlab and IEEE arithmetic. Matlab allows vector operations, thus the listings of the algorithms became shorter. We use the same example (15 × 10 Hilbert matrix) though machine precision is now $2.2204e-16$ smaller than $mc = 0.5 \times 10^{-11}$ used on the decimal HP 9825 machine.

To test the algorithms we use the test-program

9.1 Classical Schmidt

In a first step we vectorize the scalar product and work with entire column vectors. The algorithm becomes in Matlab:

```
function [Q,R] = clgs(A)
% CLGS classical Gram-Schmidt orthogonalization
[m,n] = size(A); R = zeros(n);
Q=A;
for k = 1:n,
    for i = 1:k-1,
        R(i,k) = Q(:,i)'*Q(:,k); % remove for
    end % modified-Gram-Schmidt
    Q(:,k) = Q(:,k)-R(i,k)*Q(:,i);
    R(k,k) = norm(Q(:,k)); Q(:,k) = Q(:,k)/R(k,k);
end
```

We obtain with

```
> A = hilb(15); A = A(:,1:10);
> [Q,R] = clgs(A); pruef(A,Q,R)
```

the results (we use $|A| := \max_{ij} |a_{ij}|$). Orthogonality of $Q$ is lost completely, however, $QR$ represents $A$ well to machine precision.

```
i) |A - QR| = 2.7756e-17
  ii) |Q^T Q - I| = 9.9998e-01
  iii) |Q^T A - R| = 1.6319e-05
  iv) |AR^{-1} - Q| = 3.5936e-09
```

Notice that we can even more vectorize by replacing the inner loop

```
function [Q,R] = clgsa(A)
% CLGSA classical Gram-Schmidt orthogonalization
% vectorized version
[m,n] = size(A); R = zeros(n);
Q=A;
for k = 1:n,
    R(1:k-1,k) = Q(:,1:k-1)'*Q(:,k);
end
```

```
Q(:,k) = Q(:,k) - Q(:,1:k-1)*R(1:k-1,k);
R(k,k) = norm(Q(:,k)); Q(:,k) = Q(:,k)/R(k,k);
end

Now we get with

```matlab
> A = hilb(15); A = A(:,1:10);
> [Q,R] = clgsa(A); pruef(A,Q,R)
```

i) $|A - QR| = 2.7755e-17$

ii) $|Q^TQ - I| = 9.9998e-01$

iii) $|Q^TA - R| = 1.6052e-05$

iv) $|AR^{-1} - Q| = 1.1089e-09$

### 9.2 Modified Gram-Schmidt

Both versions, modified Gram-Schmidt and the variant of Schwarz-Rutishauser should give the same results. However, the processors in use today compute with 80-bit registers and store 64-bit floating point numbers according to the IEEE standard. Therefore using Matlab vector operations or programming with for-loops yield different results.

With the variant of modified Gram-Schmidt according to Schwarz-Rutishauser

```matlab
function [Q,R] = modgs(A)
% MODGS modified Gram-Schmidt orthogonalization
% Schwarz-Rutishauser
[m,n] = size(A); R = zeros(n);
Q=A;
for k = 1:n,
    for i = 1:k-1,
        R(i,k) = Q(:,i)'*Q(:,k);
        Q(:,k) = Q(:,k) - R(i,k)*Q(:,i);
    end
    R(k,k) = norm(Q(:,k)); Q(:,k) = Q(:,k)/R(k,k);
end
```

we get with

```matlab
> A = hilb(15); A = A(:,1:10);
> [Q,R] = modgs(A); pruef(A,Q,R)
```

the results

i) $|A - QR| = 2.7756e-17$

ii) $|Q^TQ - I| = 9.9998e-01$

iii) $|Q^TA - R| = 1.6052e-05$

iv) $|AR^{-1} - Q| = 1.1089e-09$

With the second variant of modified Gram Schmidt

```matlab
function [Q,R] = modgs2(A)
% MODGS2 modified Gram-Schmidt orthogonalization
[m,n] = size(A); R = zeros(n);
for k = 1:n,
```
\[
R(k, k) = \text{norm}(A(:, k)); \\
Q(:, k) = A(:, k)/R(k, k); \\
\text{if } k < n \\
\quad R(k, k+1:n) = \text{Q}(, k)'*A(:, k+1:n); \\
\quad A(:, k+1:n) = A(:, k+1:n) - \text{Q}(, k)*R(k, k+1:n); \\
\text{end} \\
\text{end}
\]

we obtain as expected about the same results as before. \(Q\) is orthogonal to about 6 decimal digits. iv) is worse than with classical Schmidt.

\[
i) \quad |A - QR| = 5.5511e-17 \\
ii) \quad |Q^T Q - I| = 1.6957e-05 \\
iii) \quad |Q^T A - R| = 2.1317e-05 \\
iv) \quad |AR^{-1} - Q| = 2.1364e-04.
\]

9.3 Modified Gram-Schmidt with Reorthogonalization

The first variant with reorthogonalization will not work for a rank deficient matrix.

```matlab
function [Q,R] = reorthgs1(A) 
% REORTHGS1 modified Gram-Schmidt orthogonalization with 
% reorthogonalization 
% [m,n] = size(A); R = zeros(n); 
Q=A; 
for k=1:n 
\quad tt = 0; 
\quad t = norm(Q(:,k)); 
\quad reorth = 1; 
\quad while reorth, 
\quad \quad % orth: 
\quad \quad \quad for i=1:k-1 
\quad \quad \quad \quad s = Q(:,i)'*Q(:,k); 
\quad \quad \quad \quad if tt==0, R(i,k) = s; end; 
\quad \quad \quad Q(:,k) = Q(:,k)-s*Q(:,i); 
\quad \quad \quad end 
\quad \quad tt = norm(Q(:,k)); 
\quad \quad reorth=0; 
\quad \quad if tt<t/10, t=tt; reorth=1;end 
\quad end 
R(k,k) = tt; 
Q(:,k) = Q(:,k)/R(k,k); 
end
```

24
We obtain for our example a perfect orthogonality of $Q$, also identity iii) is very well fulfilled while iv) is rather poor due to the ill-conditioning of $R$.

\begin{align*}
\text{i)} & & |A - QR| & = 1.6653e-16 \\
\text{ii)} & & |Q^TQ - I| & = 1.3999e-15 \\
\text{iii)} & & |Q^TA - R| & = 1.7243e-15 \\
\text{iv)} & & |AR^{-1} - Q| & = 1.5238e-04
\end{align*}

### 9.4 Second version of modified Gram-Schmidt with reorthogonalization

This version from ritzit will work also for possibly rank deficient matrices.

```matlab
function [Q,R,z] = gramschmidt(Q)
% GRAMSCHMIDT computes the QR decomposition using
% modified Gram-Schmidt with reorthogonalization
% z is a vector that counts the reorthogonalization
% steps per column (source Ritzit)
z =[]; [m,n] = size(Q); R = zeros(n);
for k = 1:n,
    t = norm(Q(:,k));
    nach =1;
    u=0; % count variable
    while nach,
        i = 1:k-1,
        s = Q(:,i)'*Q(:,k);
        R(i,k) = R(i,k) + s;
        Q(:,k) = Q(:,k)- s*Q(:,i);
    end
    tt = norm(Q(:,k));
    if tt>10*eps*t & tt<t/10, % if short length
        nach =1; t=tt; % reorthogonalize
    else
        nach = 0;
        if tt<10*eps*t, tt=0; % linear dependent
    end
end
z =[z u];
R(k,k) = tt;
if tt*eps=0, tt =1/tt; else tt=0; end
Q(:,k) = Q(:,k)*tt;
end
```

\begin{align*}
\text{i)} & & |A - QR| & = 5.5511e-17 \\
\text{ii)} & & |Q^TQ - I| & = 1.2750e-15 \\
\text{iii)} & & |Q^TA - R| & = 1.6358e-15 \\
\text{iv)} & & |AR^{-1} - Q| & = 5.1228e-05
\end{align*}

The strength of this variant is for rank deficient matrices. Take as example
\[ H = \text{magic}(10); \]
\[ [Q, R] = \text{gramschmidt}(H) \]

We get a matrix \( Q \) with 3 zero columns and also \( R \) has 3 zero rows. So the rank 7 is well detected.

### 9.5 Matlab QR

The standard Matlab function \( \text{qr} \) with

\[ A = \text{hilb}(15); A = A(:,1:10); \]
\[ [Q, R] = \text{qr}(A, 0); \text{pruef}(A, Q, R) \]

gives the best results:

\[
\begin{align*}
    &i) \quad |A - QR| = 5.5511e-16 \\
    &ii) \quad |Q^T Q - I| = 5.5511e-16 \\
    &iii) \quad |Q^T A - R| = 2.220e-16 \\
    &iv) \quad |AR^{-1} - Q| = 2.8113e-04 \\
\end{align*}
\]

### 9.6 Implicit Householder Decomposition

The decomposition with implicit Householder transformations

\[
\begin{align*}
    \text{function} & \quad [A, d] = \text{qrhous}(A); \\
    \% \quad \text{QRHous} \quad \text{computes the QR decomposition } A = QR \\\n    \% \quad \text{using Householder transformations.} \\\n    \% \quad \text{The output matrix } A \text{ contains the Householder} \\\n    \% \quad \text{vectors } u \text{ and the upper triangle of } R. \text{ The} \\%
    \% \quad \text{diagonal of } R \text{ is stored in vector } d. \\

    [m, n] = \text{size}(A); \\
    \text{for } j = 1:n, \\
    \quad s = \text{norm}(A(j:m, j)); \\
    \quad \text{if } s == 0, \text{error('rank(A) < n'), end} \\
    \quad \text{if } A(j, j) >= 0, d(j) = -s; \text{ else } d(j) = s; \text{ end} \\
    \quad \text{fak} = \sqrt{s \cdot (s + \text{abs}(A(j, j)))}; \\
    \quad A(j, j) = A(j, j) - d(j); \\
    \quad A(j:m, j) = A(j:m, j) / \text{fak}; \\
    \quad \% \quad \text{transformation of the rest of} \\%
    \quad \% \quad \text{the matrix } G := G - u(u'G) \\
    \quad \text{if } j < n, \\
    \quad \quad A(j:m, j+1:n) = A(j:m, j+1:n) - \ldots \\
    \quad \quad \quad A(j:m, j)(A(j:m, j)'A(j:m, j+1:n)); \\
    \quad \text{end} \\
\end{align*}
\]

\[
\text{together with} \\
\text{function } z = \text{qyhous}(A, y); \\
\% \quad \text{QYHOU5: computes } z = Q y \text{ if before } Q \text{ has} \%
\% \quad \text{been computed implicitly by } A = \text{qrhous}(A) \\
\]
\[
\begin{align*}
[m, n] &= \text{size}(A); \ z = y; \\
\text{for } j = n:-1:1, \\
\quad z(j:m) &= z(j:m) - A(j:m,j) \cdot (A(j:m,j)' \cdot z(j:m)); \\
\text{end};
\end{align*}
\]
yields with the main program
\[
\begin{align*}
[m, n] &= \text{size}(A); \\
[a \ d] &= \text{qrhou}\_\text{s}(A); \\
I &= \text{eye}(m, n); \\
Q &= []; \\
\text{for } i = I, \\
\quad z &= \text{qyhou}\_\text{s}(a, i); \\
Q &= [Q \ z]; \\
\text{end} \\
R &= \text{triu}(a(1:n,:), 1) + \text{diag}(d); \\
Q &= Q(:, 1:n); \\
R &= R(1:n, 1:n); \\
\text{pruef}(A, Q, R)
\end{align*}
\]
the results:
\[
\begin{align*}
\text{i) } |A - QR| &= 9.4369e-16 \\
\text{ii) } |Q^T Q - I| &= 1.5543e-15 \\
\text{iii) } |Q^T A - R| &= 1.3323e-15 \\
\text{iv) } |AR^{-1} - Q| &= 2.8256e-04
\end{align*}
\]