CS 83

ITERATIVE REFINEMENTS OF LINEAR LEAST SQUARES SOLUTIONS BY HOUSEHOLDER TRANSFORMATIONS

BY

Å. BJÖRCK

G. GOLUB

TECHNICAL REPORT NO. CS 83

JANUARY 19, 1968

COMPUTER SCIENCE DEPARTMENT

School of Humanities and Sciences

STANFORD UNIVERSITY

Iterative Refinements of Linear Least Squares Solutions by Householder Transformations.*

Contributed by A. Björck and G. Golub.**

Theoretical background

, Let A be a given $m \times n$ real matrix with $m > n$ and of rank n and b a given vector. Let A and b be partitioned

$$A = \begin{pmatrix} A_1 \\ \hline A_2 \end{pmatrix} \begin{matrix} \} m_1 \times n \\ \} m_2 \times n \end{matrix} \quad , \quad b = \begin{pmatrix} b_1 \\ \hline b_2 \end{pmatrix}$$

where $m_1 \leq n$ and assume that A_1 has rank m_1. We wish to determine a vector x subject to the linear constraints

$$A_1 x = b_1$$

such that

$$\|r_2\| = \min. \quad , \quad r_2 = b_2 - A_2 x \; ,$$

where $\| \dots \|$ indicates the euclidian norm.

*This work was supported by the Swedish Natural Science Research Council.
**The work of this author was in part supported by NSF and ONR.

Using Lagrange multipliers it is easily shown that the solution satisfies the system of equations

$$\left(\begin{array}{c|c|c} 0 & 0 & A_1 \\ \hline 0 & I & A_2 \\ \hline A_1^I & A_2^I & 0 \end{array}\right)\left(\begin{array}{c} \lambda \\ \hline r_2 \\ \hline x \end{array}\right) = \left(\begin{array}{c} b_1 \\ \hline b_2 \\ \hline c \end{array}\right) \qquad (1)$$

where λ is the vector of Lagrange parameters and $c = 0$. For reasons which later will become evident we develop a method for solving (1) which works for an arbitrary vector c .

Let P be a permutation matrix which permutes the columns of A so that

$$AP = \left(\begin{array}{c} A_1' \\ \hline A_2' \end{array}\right) = \left(\begin{array}{c|c} A_{11}' & A_{12}' \\ \hline A_{21}' & A_{22}' \end{array}\right)$$

where A'_{11} is square and nonsingular. We now determine an orthogonal matrix Q_{11} so that

$$Q_{11}A_1' = (R_{11} \mid R_{12}) \ , \qquad (2)$$

where R_{11} is $m_1 \times m_1$ and upper triangular. Next we put

$$Q_{12} = R_{11}^{-T} A_{21}'^{T} \quad , \quad A_{22} = A_{22}' - Q_{12}^T R_{12} \qquad (3)$$

and determine an orthogonal transformation Q_{22} so that

$$Q_{22}A_{22} = \begin{pmatrix} R_{22} \\ \hline 0 \end{pmatrix} \}\ (m-n) \times n \tag{4}$$

where again R_{22} is upper triangular. Denote by R the $n \times n$ upper triangular matrix

$$R = \left(\begin{array}{c|c} R_{11} & R_{12} \\ \hline 0 & R_{22} \end{array}\right)$$

Then it is easily verified that

$$APR^{-1} = \left(\begin{array}{c|c} Q_{11} & Q_{12} \\ \hline 0 & \tilde{Q}_{22} \end{array}\right)^T \tag{5}$$

where

$$Q_{22} = (I_{n-m_1} | 0)\, Q_{22} \tag{6}$$

and I_{n-m_1} is an $(n-m_1) \times (n-m_1)$ unit matrix. Thus if we define the-vectors

$$y = \begin{pmatrix} y_1 \\ \hline y_2 \end{pmatrix} \begin{matrix} \}\ m_1 \\ \}\ n-m_1 \end{matrix} \quad , \quad d = \begin{pmatrix} d_1 \\ \hline d_2 \end{pmatrix} \begin{matrix} \}\ m_1 \\ \}\ n-m_1 \end{matrix}$$

by the relations

$$x = PR^{-1}y \quad , \quad d = PR^{-T}c \tag{7}$$

then (1) can be written

$$\left(\begin{array}{cc|cc} 0 & 0 & Q_{11}^T & 0 \\ 0 & I & Q_{12}^T & \tilde{Q}_{22}^T \\ \hline Q_{11} & Q_{12} & 0 & 0 \\ 0 & \tilde{Q}_{22} & 0 & 0 \end{array}\right) \begin{pmatrix} \lambda \\ r_2 \\ \hline -y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \hline d_1 \\ d_2 \end{pmatrix} \tag{8}$$

Using the orthogonality of Q_{11} and Q_{22} we get the following algorithm for solving (1):

$$y_1 = Q_{11}b_1$$

$$g = Q_{22}(b_2 - Q_{12}^T y_1) = \begin{matrix} g_1 \\ \hline g_2 \end{matrix} \begin{matrix} \}(n-m_1) \\ \}(m-u) \end{matrix}$$

$$y_2 = g_1 - d_2$$

$$r_2 = Q_{22}^T \begin{pmatrix} d_2 \\ \hline g_2 \end{pmatrix}$$

$$\lambda = Q_{11}^T(d_1 - Q_{12}r_2) \ .$$

Here d is defined from (7) which is also used for computing x .

A very effective method to realize the decompositions (2) and (4) is via Householder transformations [4]. Let $A' = A^{(1)}$, and let $A^{(k+1)}$, $k=1,2,\dots, n$, $k \neq m_1$ be defined as follows

$$A^{(k+1)} = P^{(k)} A^{(k)} .$$

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

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

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

$$a_{i,k}^{(k+1)} = 0 \quad , \quad i = k+1, \dots, m(k) ,$$

$$m(k) = \begin{cases} m_1 , & k < m_1 \\ m , & k > m_1 \end{cases}$$

It follows that

$$A_1^{(m_1)} = (R_{11} \mid R_{12})$$

and if we finally define

$$A^{(m_1+1)} = \left(\begin{array}{c|c} R_{11} & R_{12} \\ \hline 0 & A_{22} \end{array}\right)$$

where A_{22} is computed from (3 then $A^{(n+1)} = \left(\frac{R}{0}\right)$

It can be shown cf. [5] that $P^{(k)}$ is generated as follows:

$$\sigma_k = (\sum_{i=k}^{m(k)} (a_{ik}^{(k)})^2)^{1/2}$$

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

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

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

$$u_1^{(k)} = a_{ik}^{(k)} \quad \text{for } k < i \leq m(k) .$$

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

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

where

$$y_k^T = \beta_k u^{(k)T} A^{(k)} .$$

In computing the vector y_k and $A^{(k+1)}$, one takes advantage of the zero components of $u^{(k)}$.

The permutation of the columns of A to obtain A' = AP is conveniently done at the same time. At the kth stage the column is chosen which will maximize $|a_{kk}^{(k+1)}|$. This will ensure that the matrix R_{11} is non-singular. Let

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

Then since $|a_{kk}^{(k+1)}| = \sigma_k$, one should choose that column for which $s_j^{(k)}$ is maximized. After $A^{(k+1)}$, $k \neq m_1$, has been computed, one can compute $s_j^{(k+1)}$ as follows:

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

since the orthogonal transformations leave the column lengths invariant.

Because of the influence of rounding errors the first computed solution may not be sufficiently accurate in an ill-conditioned case. Provided the columns of A are not almost linearly dependent to working accuracy, the solution may be improved by the following iterative procedure. Put

$$B = \left(\begin{array}{c|c|c} 0 & 0 & A_1 \\ \hline 0 & I & A_2 \\ \hline A_1^T & A_2^T & 0 \end{array}\right) , \quad z = \left(\begin{array}{c} \lambda \\ \hline r_2 \\ \hline x \end{array}\right) , \quad h = \left(\begin{array}{c} b_1 \\ \hline b_2 \\ \hline 0 \end{array}\right) .$$

and let $z^{(0)} = 0$. The s^{th} iteration involves the three steps:

$$\text{(i)} \quad f^{(s)} = h - Bz^{(s)} \quad ,$$

$$\text{(ii)} \quad \delta z^{(s)} = B^{-1} f^{(s)} \quad ,$$

$$\text{(iii)} \quad z^{(s+1)} = z^{(s)} + \delta z^{(s)} \quad .$$

It is essential that the residuals $f^{(s)}$ are computed using double precision accumulation of inner-products. We then solve for $\delta z^{(s)}$ by the method developed above, using the same decompostion in all iterations. Note that $f^{(s)}$ generally differs from zero also in the last n components, which explains why we did not assume c=0 in (1).

It has been shown in [1] that if the iterations 'converge' then for sufficiently large s the accuracy in $(z^{(s)} + \delta z^{(s)})$ will be approximately the same as if double precision had been used throughout without refinement.

Let the number of operations needed for the decomposition resp. one iteration step for a single right hand side be N_d resp. N_s . Then a simple calculation shows that

$$N_d = \left[n^2(m - \frac{n}{3}) - m_1m_2(n - \frac{m_1}{2})\right] (1 + O(\frac{1}{n}))\ \text{s.p.}$$

$$N_s = \left[4n(m - \frac{n}{4}) - 2m_1m_2\right] (1 + O(\frac{1}{n}))\ \text{s.p.} + 2mn\ \text{d.p.}$$

where s.p. refers to single precision operations and d.p. to operations performed with double precision accumulation

Applicability

The algorithm least squares solution may be used to compute accurate solutions and residuals to linear least squares problems with or without linear constraints. It may also be used to compute accurate solutions to systems of linear equations where A is a square matrix and to compute accurate inverses of such matrices. The procedure will fail when A_1 or A modified by rounding errors has rank less than m_1 or n respectively. It will also fail if A is so ill-conditioned that there is no perceptible improvement in the iterative refinement. The matrix A is retained in order to form the residuals. When $m >> n$ the large storage requirement of this procedure might make it preferable to use instead a double precision version of the Householder decomposition without iterative refinement. Note that in the linear equation case the calculation of residuals may be suppressed by putting $m_1 = m = n$.

Formal parameter list

Input to procedure least squares

m_1	number of linear constraints $m_1 \leq n$.
m	total number of equations
n	number of unknowns $n < m$
a	an $m \times (n+1)$ array having the given matrix as first n columns
p	number of right hand sides
b	an $m \times p$ array containing the given right hand sides
eta	the largest number for which $1 + eta = 1$ on the computer
singular	exit used when A_1 or A modified by rounding errors has rank less than m_1 or n respectively
fail	exit used when the iterative refinement fails to improve the solution

Output of procedure <u>least squares</u>

x	an $n\times p$ array consisting of the p solution vectors
res	an $m_2\times p$ array consisting of the p residuals vectors

<u>ALGOL Program</u>

```
procedure least squares solution(ml) data: (m,n,a,p,b,eta) failure
          exits: (singular, fail) result: (x, res);
value ml,m,n,p,eta;
integer ml,m,n,p; real eta;
array a,b,x,res; label fail, singular;
comment The array a[1:m,1:n+1] contains in its first n columns the
        given matrix of an overdetermined system of m linear equations
        in n unknowns (m ≥ n), where the first ml equations
        (ml ≤ n) are to be strictly satisfied. For the p right
        hand sides given as columns of the array b[1:m,1:p] the
        least squares solution and the residuals are computed and
        stored in the columns of the arrays x[1:n,1:p] and
        res[ml+1:m,1:p] respectively. If rank(a) < n or
        rank(a1) < ml the emergency exit singular is used. If the
        iterative refinement fails to improve the solution sufficiently
        the exit fail is used. In either case b and the first n
        columns of a are left intact. The (n+1)st column in
        a is used as temporary storage for the sucessive right hand
        sides. Eta is the relative machine precision;
```

```
begin integer i,j,ℓ;
      array xℓ[1:n+1], resℓ[1:m], alpha[1:n], qr[0:m,1:n];
      integer array pivot[1:n];
      real procedure innerprod(i,m,n,ai,bi,c);
      value m,n,c;
      real ai,bi,c; integer i,m,n;
      begin real sum;
            sum:-0;
            for i:=m step 1 until n do sum:=sum+ai×bi;
            innerprod:=sum+c
      end innerprod;
      real procedure innerproddp(i,m,n,ai,bi,c);
      value m,n,c
      real ai,bi,c; integer i,m,n;
      comment This procedure accumulates the sum of products ai×bi
              and adds it to the initial value c in double precision.
              The body cf this procedure cannot be expressed in ALGOL.
              begin real s1,s2, (s1,s2):=0,
                    for i:=m step 1 until n do
                    (s1,s2):=(s1,s2)+ai×bi,  comment dbl.pr.acc.
                    innerproddp:=((s1,s2)+c) rounded
              end innerproddp;
      procedure decompose(m1)data:(m,n,eta) data and result:(qr)
                result: (alpha,pivot) failure exit:(singular);
      value m1,m,n,eta;
      integer m1,m,n; real eta; array qr, alpha;
```

```
        integer array pivot; label singular;
    comment Decompose uses essentially a sequence of elementary orthogonal
            transformations  (I - beta u u^T)  to determine a qr-decomposition
            of the matrix given in the array  qr[1:m,1:n] .  The diagonal
            elements of the upper triangular matrix r  are stored in the
            array  alpha[1:n],  the offdiagonal elements in the upper
            right triangular part of qr.  The nonzero components 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,mr,s; boolean fsum;
        real beta,sigma,alphak,qrkk,smax,y;array sum[1:n];
        mr:= ml; fsum:= true;
        for j:=1 step 1 until n do pivot[j]:=j;
        for k:=1 step 1 until n do
        begin comment k-th hpuseholder transformation;
            if k=ml+1 then
            begin fsum:=true; mr:=m end;
            if fsum then
        piv: for j:=k step 1 until n do
            sum[j]:=innerprod(i,k,mr,qr[i,j], qr[i,j], 0);
            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 fsum then smax:=sigma; fsum:=sigma < eta×smax;
if fsum then goto piv;
if jbar ≠ k then
begin comment column interchange;
      i:=pivot[k]; pivot[k]:=pivot[jbar]; pivot[jbar]:=i;
      sum[jbar]:=sum[k];
      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;
sum[k]:=sigma:=innerprod(i,k,mr,qr[i,k], qr[i,k], 0);
if sigma = 0 then goto singular;
grkk:=qr[k,k]; alphak:=alpha[k]:=
if qrkk < 0 then sqrt(sigma) else -sqrt(sigma);
qr[k,k]:=qrkk-alphak;
beta:=qr[0,k]:=alphak×qr[k,k];
for j:=k+1 step 1 until n do
begin y:=innerprod(i,k,mr,qr[i,k], qr[i,j], 0)/beta;
      for i:=k step 1 until mr do qr[i,j]:=qr[i,j]+y×qr[i,k];
      sum[j]:=sum[j] - qr[k,j]↑2
end j;
if k-ml then
for j:=ml+1 step 1 until m do
```

```
                for s:=1 step 1 until n do
                begin mr:= if s>m1 then m1 else s-1;
                      y:=-innerprod(i,1,mr,qr[i,s],qr[j,i],-qr[j,s]);
                      qr[j,s]:= if s>m1 then y else y/alpha[s]
                end s
            end k-th householder-transformation
        end decompose;
        procedure accsolve(m1)data:(m,n,a,qr,alpha,pivot,eta) result:(x,res)
                failure exit:(fail);
        value m1,m,n,eta;
        integer m1,m,n; real eta; array a,qr,alpha,x,res;
        integer array pivot; label fail;
        comment Accsolve uses the decomposition of a stored in the array
             qr[1:m,1:n] by decompose for the iterative refinement of the
             least squares solution. The right hand side b is given in
             the (n+1)st column of the array a[1:m,1:n+1]. The
             residuals of the augmented system of (m+n) equations are
             computed using the procedure innerproddp which forms accurate
             inner-products. As initial approximation is taken x=r=0,
             and the two first iterations are always executed. The
             iterations are repeated as long as the norm of the correction
             at any stage is less than 1/8 of that at the previous stage
             until the norm of the correction is less than epsilon times
             the norm of the solution. Exit to label fail is made if the
             solution fails to improve sufficiently;
```

```
begin integer i,j,k,s;
        real c,nx,nr,ndx1,ndx2,ndr1,ndr2,eta2;
        array f[1:m], g[1:n];
        procedure householder(p,q,r,m);
        value p,q,r,m; integer p,q,r,m;
        for s:=p step q until r dc
        begin  :=innerprod(i,s,m,qr[i,s], f[i], 0)/qr[0,s];
                for i:=s step 1 until m do f[i]:=f[i] + cXqr[i,s]
        end householder;
        eta2:=(eta↑2)↑2; x[n+1]:=-1;
        comment initial values;
        for j:=1 step 1 untiln do x[j]:=g[j]:=0;
        for i:=1 step 1 until m do
        begin res[i]:=0; f[i]:=a[i,n+1] end
        for k:=0,1,k+1 while (64×ndx2 < ndxl A ndx2 > eta2×nx) V
        (64×ndr2 < ndrl ∧ ndr2 > eta2×nr) dc
        begin comment k-th iteration step;
                ndx1:=ndx2; ndr1:=ndr2; ndx2:=ndr2:=0;
                if k ≠ 0 then
                begin comment-new residuals;
                        for i:=1 step 1 until m do res[i]:=res[i] + f[i];
                        for s:=1 step 1 until n do
                        begin j:=pivot[s]; x[j]:=x[j] + g[s];
                                g[s]:=-innerproddp(i,1,m,a[i,j], res[i], 0);
                                g[s]:=-innerprod(i,1,s-1,qr[i,s], g[i], -g[s])/
                                alpha[s]
```

```
                    end;
                    for i:=1 step 1 until m do
                    f[i]:=-innerproddp(j,1,n+1,a[i,j], x[j],
                    if i > m1 then res[i] else 0)
            end new residuals;
            householder(1,1,m1,m1);
            for i:=m1+1 step 1 until m do
            f[i]:=-innerprod(s,1,m1,qr[i,s], f[s], -f[i]);
            householder(m1+1,1,n,m);
            for i:=1 step 1 until n do
            begin c:=f[i]; f[i]:=g[i];
                g[i]:=if i>m1 then c-g[i] else c
            end;
            for s:-n step -1 until 1 do
            begin g[s]:=innerprod(i,s+1,n,qr[s,i], g[i], -g[s])/
                alpha[s]; ndx2:=ndx2+g[s]↑2
            end;
            householder(n,-1,m1+1,m);
            for s:=1 step 1 until m1 do
            f[s]:=-innerprod(i,m1+1,m,qr[i,s], f[i], -f[s]);
            householder(m1,-1,1,m1);
            for i:=1 step 1 until m do
            ndr2:=ndr2+f[i]↑2;
            if k = 0 then begin nx:=ndx2; nr:=ndr2 end
        end k-th iteration step;
            if ndr2 > eta2×nr Λ ndx2 > eta2×nx then goto fail
        end accsolve;
        for j:=1 step 1 until n do
        for i:=1 step 1 until m do qr[i,j]:=a[i,j];
        decompose(m1,m,n,eta,qr,alpha,pivot,singular);
        for ℓ:=1 step 1 until p do
        begin comment ℓ-th right hand side;
              for i:=1 step 1 until m do a[i,n+1];=b[i,ℓ];
              accsolve(m1,m,n,a,qr,alpha,pivot,eta,xℓ,resℓ,fail);
              for j:=1 step 1 until n do x[j,ℓ]:=xℓ[j];
              for i:=m1+1 step 1 until m do res[i,ℓ]:=resℓ[i]
        end ℓ-th right hand side
    end least squares;
```

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 $u^{(k)}$ and the derived matrix Q_{12} are stored on and below the leading diagonal of qr . The diagonal elements of R, the reduced matrix, are stored in the array α, and the elements β_k on row number zero in qr .

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.

The recursive computation of $s_j^{(k)}$ will fail if A is sufficiently ill-conditioned. To prevent this $s_j^{(k)}$ are recomputed every time the condition

$$\max_{k\le j\le n} s_j^{(k)} < \eta \cdot \max_{k'\le j\le n} s_j^{(k')}$$

is satisfied, where k' is the last step at which this was done. Since the number of iterations needed is dependent on the right hand side the iterative refinement is executed for one right hand side at a time. During the refinement the current right hand side is transferred to the $(n+1)$st column of A .

In accsolve the first set of solutions is taken to be null vectors, and the two first iteration steps are always executed. The iteration for the current right hand side is terminated when the conditions (i) and (ii) below are simultaneously satisfied:

(i) $\|\delta x^{(s)}\|_2 > 0.125\|\delta x^{(s-1)}\|_2$ or $\|\delta x^{(s)}\|_2 \le \eta\|x^{(1)}\|_2$

(ii) $\|\delta r^{(s)}\|_2 \ge 0.125\|\delta r^{(s-1)}\|_2$ or $\|\delta r^{(s)}\|_2 \le \eta\|r^{(1)}\|_2$.

If the iteration has been terminated and at the same time

$$\|\delta x^{(s)}\|_2 > 2\eta\|x^{(1)}\|_2 \quad \text{and} \quad \|\delta r^{(s)}\|_2 > 2\eta\|r^{(1)}\|_2 ,$$

then the exit fail is used.

Both a single precision and a double precision inner product routine are used. On a computer where double precision accumulation of inner products is fast, the double precision routine can be used throughout,

Discussion of Numerical Properties

The procedure has been analyzed in [1] for $m_1 = 0$ under the assumption that all inner-products are accumulated in double precision. (If single precision inner-products are used where possible, the bounds given below for the rate of convergence and the error will increase by a factor less than m .)

Let t_1 and t_2 be the number of binary digits in our single and double precision floating point mantissas. Put

$$\alpha = 32.6\, n^{3/2}\, 2^{-t_1} \kappa(A)$$

where

$$\kappa(A) = \max_{\|x\|_2=1} \|Ax\|_2 \Big/ \max_{\|x\|_2=1} \|Ax\|_2 ,$$

and assume that $\alpha < 1$. If the errors made in computing the residuals and in adding the corrections can be neglected, then

$$\begin{pmatrix} \|r-r^{(s)}\|_2 \\ \|A\|_2\|x-x^{(s)}\|_2 \end{pmatrix} < 14.4n^{3/2}2^{-t_1}\rho^{s-1} \begin{pmatrix} \kappa'+\frac{4}{3} & \frac{5}{3} \\ \kappa'(\kappa'+\frac{4}{3}) & \kappa'\frac{5}{3} \end{pmatrix} \begin{pmatrix} \|r\|_2 \\ \|A\|_2\|x\|_2 \end{pmatrix}$$

where

$$\kappa' = (1-\alpha)^{-1/2}\kappa(A) ,$$

and the "initial rate of convergence' ρ is bounded by

$$\rho < 38.7\, n^{3/2}(\kappa'+\frac{1}{2})\, 2^{-t_1} .$$

The process 'converges' if $\rho < 1$. Then for sufficiently large s the errors will satisfy

$$\begin{pmatrix} \|r-r^{(s)}\|_2 \\ \|A\|_2\|x-x^{(s)}\|_2 \end{pmatrix} < (1-\rho)^{-1}K \begin{pmatrix} 1 \\ \kappa' \end{pmatrix} + 2^{-t_1} \begin{pmatrix} \|r\|_2 \\ \|A\|_2\|x\|_2 \end{pmatrix},$$

where-

$$K = 14.4\, n^{3/2}\, 2^{-2t_1}((\kappa'+\frac{4}{3})\, \|r\|_2 + \frac{5}{3}\, \|A\|_2\|x\|_2) +$$

$$1{,}022\, 2^{-t_2}(\kappa'(m+4)\, \|r\|_2 + (n+5)\, \|A\|_2\|x\|_2) .$$

If $t_2 \geq 2t_1$ then the first term in K usually dominates, and $x^{(s)} + \delta x^{(s)}$ will ultimately have t_1 more correct binary digits than $x^{(1)}$. Note however that the process may well converge even if $x^{(1)}$ has relative error greater than 1. To get full benefit of the refinement we ought to have $t_2 \approx 2t_1$, but there is nothing to be gained by taking t_2 much greater than $2t_1$.

Since it is possible to have $x = 0$ or $r = 0$, it is obvious that even when $\rho < 1$, we cannot guarantee that $x^{(s)}$ or $r^{(s)}$ ultimately will have a small relative error. Let

$$\gamma = (\kappa' + \frac{4}{3}) \frac{\|r\|_2}{\|A\|_2 \|x\|_2}$$

and assume that $\rho < 1/4$ and that the second term in K can be neglected. If

$$\gamma < 1.58 \frac{1}{\rho}$$

then we will ultimately have

$$\|x - x^{(s)}\|_2 < 2.2^{-t_1} \|x\|_2 .$$

Similarly if

$$1,61\rho < \gamma$$

then ultimately

$$\|r - r^{(s)}\|_2 < 2 \cdot 2^{-t_1} \|r\|_2 .$$

Note that $r^{(s)}$ will converge to the exact residual corresponding to the correct solution x. When $\|r\| \ll \|A\| \|x\|$ these may be very different from the residual corresponding to x rounded to single precision. In many cases the later may be the more relevant.

Test Results

The procedure was tested on the CD 3600 (University of Uppsala) which, has $t_1 = 36$ and $t_2 = 84$, with $\eta = 2^{-36} \approx 1.5 \; 10^{-11}$. The matrix A consists of the last six columns of the inverse of the 8 x 8 Hilbert matrix. For $m_1 = 0$ two right hand sides were treated. The: first, b_1, is chosen so that the system $Ax = b_1$ is compatible i.e. $r = 0$. The second, b_2, is obtained by adding to b_1 a vector orthogonal to the columns of A, the length of which was adjusted so that

$$(\kappa' + \frac{4}{3}) \frac{\|r\|_2}{\|A\|_2 \|x\|_2} \approx \frac{1}{2} \cdot 10^6 .$$

Thus in both cases the exact solution is the same, namely

$$x = (1/3, \; 1/4, 1/5, 1/6, 1/7, 1/8)^T .$$

Due to the large residuals in the second case however, this system is much more ill-conditioned cf. [1]. For $m_1 = 2$ the same matrix A and the right hand sides b_1 and b_3 was used where b_3 was obtained by changing b_2 in its first two components so that the exact solution x remains the same. Note that all problems are so ill-conditioned that $t_1 \geq 32$ is required for convergence.

The results for $m_1 = 0$ confirms that the "initial rate of convergence" is independent of the right hand side. In fact (disregarding the first step) the errors in the components of x and r decreases initially with a factor approximately equal to 10^{-3}. For economy of presentation, we have given only the last six components of $r^{(s)}$; the behavior of the other components is exactly analogous. For the right hand side b_1, $x^{(4)}$ is already correct to working accuracy. The iteration is terminated after the computation of $\delta x^{(5)}$ and $\delta r^{(5)}$ when the condition $\|\delta r^{(5)}\|_2 < \eta\|r^{(1)}\|_2$ is satisfied. For the right hand side b_2, $x^{(1)}$ is in error by a factor almost equal to 10^3 ! The iteration is again terminated after $\delta x^{(5)}$ and $x^{(5)}$ is correct to working accuracy. This accuracy which seems to be more than could be expected is explained by the fact that the residuals (b_2-Ax) are integers which can be represented exactly in the machine. In fact $r^{(s)}$ exactly equals r for $s \geq 4$ which makes the problem no more ill-conditioned when $s > 4$ than for the r.h.s. b_1.

The behavior when $m_1 = 2$ is exactly analogous. Note however that the rate of convergence is faster almost by a factor of 10^2 compared to the case $m_1 = 0$. For the right hand sides b_1 and b_3

five respectively four steps of the iteration are executed. For b_1, already $x^{(3)}$ is correct to working accuracy and for b_3, $x^{(4)}$ is almost correct.

Example

A						B		
20160	-92400	221760	-288288	132192	-51480	945	8400945	945
-952560	4656960	-11642400	15567552	-10594584	2882880	-40320	4159680	-40320
11430720	-58212000	149688000	-204324120	141261120	-38918880	456120	3256120	3256120
-58212000	304920000	-800415000	1109908800	-776936160	216216000	-2236080	-136080	-136080
149688000	-800415000	2134440000	-2996753760	2118916800	-594594000	5599440	7279440	7279440
-204324120	1109908800	-2996753760	4249941696	-3030051024	856215360	-7495488	-6095488	-6095488
141261120	-776936160	2118916800	-3030051024	2175421248	-618377760	5105100	6305100	6805100
-38918880	216216000	-594594000	856215360	-618377760	176679360	-1389960	-339960	-339960

$m_1 = 0$, rhs b_1 $\qquad$ $x^{(s)}$, $s = 1,2,3,4.$

$3.33323\ 25269_{10}-1$	$2.49983\ 36160_{10}-1$	$1.99979\ 68894_{10}-1$	$1.66644\ 47493_{10}-1$	$1.42834\ 14088_{10}-1$	$1.24976\ 82424_{10}-1$
$3.33333\ 35247_{10}-1$	$2.50000\ 03124_{10}-1$	$2.00000\ 03809_{10}-1$	$1.66666\ 70842_{10}-1$	$1.42857\ 18638_{10}-1$	$1.25000\ 04414_{10}-1$
$3.33333\ 33334_{10}-1$	$2.50000\ 00001_{10}-1$	$2.00000\ 00001_{10}-1$	$1.66666\ 66668_{10}-1$	$1.42857\ 14287_{10}-1$	$1.25000\ 00001_{10}-1$
$3.33333\ 33334_{10}-1$	$2.50000\ 00000_{10}-1$	$2.00000\ 00000_{10}-1$	$1.66666\ 66667_{10}-1$	$1.42857\ 14286_{10}-1$	$1.25000\ 00000_{10}-1$

$r^{(s)}$, $s=1,2,3,4,5.$

$9.32626\ 24303_{10}-05$	$2.12487\ 75112_{10}-04$	$2.46137\ 01049_{10}-04$	$2.50505\ 06480_{10}-04$	$2.43935\ 32104_{10}-04$	$2.33348\ 99208_{10}-04$
$5.05114\ 03416_{10}-07$	$4\ 58496\ 31646_{10}-08$	$-2.65944\ 32257_{10}-07$	$-4.67148\ 08377_{10}-07$	$-5.95081\ 64441_{10}-07$	$-6.75178\ 33458_{10}-07$
$3.65217\ 71718_{10}-11$	$-1.25607\ 41336_{10}-10$	$-2.09271\ 44739_{10}-10$	$-2.52288\ 19034_{10}-10$	$-2.73216\ 93519_{10}-10$	$-2.81694\ 14023_{10}-10$
$1.95300\ 70174_{10}-13$	$-2.68782\ 76487_{10}-14$	$9.11843\ 16499_{10}-14$	$1.68927\ 42630_{10}-13$	$2.19246\ 26557_{10}-13$	$2.51362\ 10252_{10}-13$
$4.37027\ 04750_{10}-15$	$-7.43763\ 51683_{10}-15$	$-8.15470\ 18418_{10}-15$	$-8.40092\ 32237_{10}-15$	$-7.77909\ 61723_{10}-15$	$-7.37817\ 06343_{10}-15$

m_1 o rhs b_2

$x^{(s)}$, s = 1,2,3 4 5.

$5.56239\ 01547_{10}{+1}$	$9.02800\ 72549_{10}{+1}$	$1.09708\ 61620_{10}{+2}$	$1.19973\ 52027_{10}{+2}$	$1.24798\ 36748_{10}{+2}$	$1.26358\ 19430_{10}{+2}$
$3.37777\ 18060_{10}{-1}$	$2.56809\ 75057_{10}{-1}$	$2.07823\ 90423_{10}{-1}$	$1.74786\ 46338_{10}{-1}$	$1.50905\ 26268_{10}{-1}$	$1.32794\ 36156_{10}{-1}$
$3.33311\ 57908_{10}{-1}$	$2.49964\ 54446_{10}{-1}$	$1.99956\ 83846_{10}{-1}$	$1.66619\ 41295_{10}{-1}$	$1.42807\ 94663_{10}{-1}$	$1.24950\ 15446_{10}{-1}$
$3.33333\ 33117_{10}{-1}$	$2.49999\ 99664_{10}{-1}$	$1.99999\ 99609_{10}{-1}$	$1.66666\ 66257_{10}{-1}$	$1.42857\ 13875_{10}{-1}$	$1.24999\ 99598_{10}{-1}$
$3.33333\ 33334_{10}{-1}$	$2.50000\ 00000_{10}{-1}$	$2.00000\ 00000_{10}{-1}$	$1.66666\ 66667_{10}{-1}$	$1.42857\ 14286_{10}{-1}$	$1.25000\ 00000_{10}{-1}$

$r^{(s)}$, s = 1,2,3,4.

$2.80130\ 68864_{10}{+6}$	$2.09994\ 38069_{10}{+6}$	$1.67905\ 65883_{10}{+6}$	$1.39850\ 07847_{10}{+6}$	$1.19815\ 74912_{10}{+6}$	$1.04794\ 99318_{10}{+6}$
$2.79999\ 98248_{10}{+6}$	$2.09999\ 96497_{10}{+6}$	$1.67999\ 96002_{10}{+6}$	$1.39999\ 95972_{10}{+6}$	$1.19999\ 96122_{10}{+6}$	$1.04999\ 96339_{10}{+6}$
$2.79999\ 99995_{10}{+6}$	$2.10000\ 00000_{10}{+6}$	$1.68000\ 00003_{10}{+6}$	$1.40000\ 00006_{10}{+6}$	$1.20000\ 00007_{10}{+6}$	$1.05000\ 00008_{10}{+6}$
$2.80000\ 00000_{10}{+6}$	$2.10000\ 00000_{10}{+6}$	$1.68000\ 00000_{10}{+6}$	$1.40000\ 00000_{10}{+6}$	$1.20000\ 00000_{10}{+6}$	$1.05000\ 00000_{10}{+6}$

m_1 = 2, rhs b_1

$x^{(s)}$, s = 1,2,3.

$3.33325\ 69325_{10}{-1}$	$2.49985\ 84680_{10}{-1}$	$1.99981\ 23581_{10}{-1}$	$1.66644\ 84069_{10}{-1}$	$1.42833\ 37432_{10}{-1}$	$1.24975\ 07088_{10}{-1}$
$3.33333\ 33323_{10}{-1}$	$2.49999\ 99994_{10}{-1}$	$2.00000\ 00002_{10}{-1}$	$1.66666\ 66679_{10}{-1}$	$1.42857\ 14308_{10}{-1}$	$1.25000\ 00030_{10}{-1}$
$3.33333\ 33334_{10}{-1}$	$2.50000\ 00000_{10}{-1}$	$2.00000\ 00000_{10}{-1}$	$1.66666\ 66667_{10}{-1}$	$1.42857\ 14286_{10}{-1}$	$1.25000\ 00000_{10}{-1}$

$r^{(s)}$, s = 1,2,3,4,5.

$2.13646\ 45736_{10}{-03}$	$1.99172\ 38410_{10}{-03}$	$-3.50968\ 45825_{10}{-03}$	$-4.06513\ 18158_{10}{-03}$	$-4.21736\ 74494_{10}{-03}$	$-4.18894\ 04056_{10}{-03}$
$-7.38044\ 15024_{10}{-08}$	$4.80542\ 04172_{10}{-07}$	$1.37349\ 84350_{10}{-07}$	$-2.95920\ 01738_{10}{-07}$	$-6.59523\ 94825_{10}{-07}$	$-9.35348\ 33922_{10}{-07}$
$-1.65462\ 21882_{10}{-10}$	$1.59782\ 75647_{10}{-10}$	$1.15019\ 08106_{10}{-10}$	$1.48955\ 70888_{10}{-11}$	$-8.07926\ 91980_{10}{-11}$	$-1.59359\ 23414_{10}{-10}$
$-4.06763\ 50023_{10}{-14}$	$4.20185\ 43185_{10}{-14}$	$4.78313\ 47444_{10}{-14}$	$3.51856\ 16491_{10}{-14}$	$1.95614\ 94080_{10}{-14}$	$5.25629\ 00592_{10}{-15}$
$-1.48834\ 87910_{10}{-14}$	$8.33620\ 08075_{10}{-15}$	$1.70696\ 27914_{10}{-14}$	$2.04140\ 44723_{10}{-14}$	$2.14757\ 15640_{10}{-14}$	$2.14979\ 05645_{10}{-14}$

$m_1 = 2$,   rhs b_3

$x^{(s)}$, $s = 1,2,3,4$.

$5.40942\ 34352_{10}-1$	$6.56346\ 90542_{10}-1$	$7.60691\ 80418_{10}-1$	$8.39395\ 19792_{10}-1$	$8.94072\ 75252_{10}-1$	$9.29507\ 22442_{10}-4$
$3.33414\ 78920_{10}-1$	$2.50155\ 63628_{10}-1$	$2.00211\ 42052_{10}-1$	$1.66917\ 52116_{10}-1$	$1.43134\ 91136_{10}-1$	$1.25295\ 50085_{10}-4$
$3.33333\ 35241_{10}-1$	$2.50000\ 03591_{10}-1$	$2.00000\ 04828_{10}-1$	$1.66666\ 72353_{10}-1$	$1.42857\ 20546_{10}-1$	$1.25000\ 06620_{10}-4$
$3.33333\ 33334_{10}-1$	$2.50000\ 00001_{10}-1$	$2.00000\ 00001_{10}-1$	$1.66666\ 66668_{10}-1$	$1.42857\ 14287_{10}-1$	$1.25000\ 00001_{10}-4$

$r^{(s)}$, $s = 1,2,3$.

$2.79993\ 42742_{10}+6$	$2.10010\ 54284_{10}+6$	$1.68006\ 97605_{10}+6$	$1.40000\ 52065_{10}+6$	$1.19994\ 83616_{10}+6$	$1.04990\ 48202_{10}+6$
$2.79999\ 99780_{10}+6$	$2.10000\ 00226_{10}+6$	$1.68000\ 00222_{10}+6$	$1.40000\ 00120_{10}+6$	$1.20000\ 00010_{10}+6$	$1.04999\ 99914_{10}+6$
$2.80000\ 00000_{10}+6$	$2.10000\ 00000_{10}+6$	$1.08000\ 00000_{10}+6$	$1.40000\ 00000_{10}+6$	$1.20000\ 00000_{10}+6$	$1.05000\ 00000_{10}+6$

References

[1] Björck, Å.: Iterative Refinement of Linear Least Squares Solutions I. BIT 7(1967), to appear.

[2] Businger P. and Golub, G. H.: Linear Least Squares Solutions by Householder Transformations. Num. Math. 7, 269-276 (1965).

[3] Golub,, G. H.: Numerical Methods for Solving Linear Least Squares Problems. Num. Math. 7, 206-216 (1965).

[4] Householder, A. S.: Unitary Triangularization of a Nonsymmetric Matrix. Assoc. Comput. Mach. 5, 339-342 (1958).

[5] Wilkinson, J. H.: Householders Method for Symmetric Matrices. Num. Math. 4, 354-361 (1962).