AN IMPLEMENTATION OF THE QR FACTORIZATION FOR SOLVING OVERDETERMINED SYSTEMS OF LINEAR EQUATIONS

L. F. ESCUDERO

Different procedures for solving an overdetermined system of linear equations are revisited and new technical details in its computer implementations are described. The new procedures produce more stable solutions than the implementation of the direct theoretical approaches.

1. INTRODUCTION AND MOTIVATION

In this paper we describe an implementation of a procedure to factorize a nxt full rank matrix A so that we may solve with the maximum accuracy an overdetermined system of linear equations. In this paper maximum accuracy means maximum accuracy in today available computers. Consider the problem

$$A\mu = g \qquad (1.1)$$

where g is an n-column vector, μ is the unknown t-column vector and n>t. If the system is not compatible, vector μ has to minimize the norm $||b||^2$

$$||b||^2 = ||g-A\mu||^2 \qquad (1.2)$$

If the system is compatible, vector μ that satisfies (1.1), say μ_L is termed minimal least square solution.

The algorithms described in this paper calculate least square solutions to overdetermined systems of linear equations through the use of modified Gram-Schmidt orthogonalization with iterative refinement. The least square problem is a very general one which arises in a variety of contexts in many scientific disciplines. Frequently, least squares problems are ill-conditioned, and not all algorithms for computing solutions are numerically stable. See Escudero /1/. Among the variety of cases in which this problem arise, there are two very important cases; one is the multiple linear regression in which the data are the n observed values (vector g) of a dependent variable and the observed values (matrix A) in the same n experiments corresponding to t independent variables; the problem consists in obtaining the vector μ of the regression coefficients such that the residual $||b||^2$ in eq. (1.2) is to be minimized. (In this case the partial regression coefficient μ_i takes the influence of independent variable, say X_i on the dependent variable, say Y).

The other important case with direct applicability of this work is the general non-linear constrained problem: min f(x) subject to $c(X) \geq 0$, where f(X) and c(X) are differenciable non-linear functions. The point X is stationary (see Escudero /1/) if $||b||^2 = 0$ in eq. (1.2), where A is the Jacobian matrix of the active constraints (c(X)=0), g is the gradient vector of objective function f(X) evaluated at point X and μ is the corresponding vector of Lagrange multipliers. At an intermediate iteration of the algorithm for solving the problem, point X needs that, at least, $||b||^2 = 0$ for satisfying the optimality conditions. If $||b||^2 \neq 0$, that means that the vector μ_L for which $||b||^2$ is minimized is the so-called first-order estimate of the Lagrange multipliers. See /1/.

The necessary and sufficient condition /3 p. 309/ for the minimal least square solution μ_L is that

$$A^t(g-A\mu) = 0 \qquad (1.3)$$

where A^t is the transpose of matrix A.

- L.F. Escudero. IBM Scientific Center. Palo Alto, California. On leave from IBM España, P° de la Castellana, 4, Madrid-1.
- Article rebut l'Abril del 1980, nova versió.

Sufficient

Suppose $A^t(g-A\mu) = 0$. The residual $r(\varepsilon)$ corresponding to any other vector $\mu+\varepsilon$, where ε is a vector, is

$$||r(\varepsilon)||^2 = ||g-A\mu-A\varepsilon||^2$$
$$= ||g-A\mu||^2 + ||A\varepsilon||^2 - 2(g-A\mu)^t A\varepsilon$$
$$= ||g-A\mu||^2 + ||A\varepsilon||^2 \qquad (1.4)$$

Since $||A\varepsilon||^2$ is non-negative, the residual $||g-A\mu||^2$ is a minimum. Unless $A\varepsilon = 0$, the residual corresponding to $\mu+\varepsilon$ is greater than the residual corresponding to μ; the only other vectors with minimum residual are those vectors $\mu+\varepsilon$ for which $A\varepsilon = 0$. But since A is a full rank matrix, $A\varepsilon \neq 0$.

Necessary

Suppose μ has a residual (1.2) that is minimum, but

$$A^t(g-A\mu) = s \neq 0 \qquad (1.5)$$

The residual corresponding to $\mu+\varepsilon s$ where ε is a scalar, is

$$||g-A\mu-A\varepsilon s||^2 = ||g-A\mu||^2 -$$
$$- 2\varepsilon s^t A^t(g-A\mu) + \varepsilon^2||As||^2 \qquad (1.6)$$

For sufficiently small positive ε, the right hand-side of (1.6) is less than $||g-A\mu||^2$; but it contradicts the hypothesis that the residual of μ is a minimum.

Then the vector μ_L that satisfies eq. (1.3) is

$$\mu_L = A^+ g \qquad (1.7)$$

where A^+ is the pseudoinverse of matrix A. There are several computational methods to calculate matrix A^+. A well known method computes A^+ such that

$$\mu_L = (A^tA)^{-1}A^t g \qquad (1.8)$$

assuming that (A^tA) is non-singular; it will be non-singular if A is a full rank matrix. But from a practical point of view even in this case (since the computer has finite precision) the rounding errors produce a solution μ_L that does not satisfy eq. (1.3).

In order to avoid the computation of matrix A^+, we may factorize matrix A such that

$$Q^tA = \begin{pmatrix} R \\ 0 \end{pmatrix} \qquad (1.9)$$

where Q^t is a nxn non-symmetric orthogonal ($Q^tQ=I$) matrix and R is a txt upper triangular matrix (not necessarily with identity diagonal). Matrix Q may be partitioned so that

$$Q = (Q_1 Q_2) \qquad (1.10)$$

where Q_1 is a nxt matrix and Q_2 is a nx(n-t) matrix, so that

$$A = Q_1 R \qquad (1.11)$$

Since A is a full rank matrix, it results

(a) R is a non-singular matrix

(b) $$I=QQ^t=(Q_1Q_2)\begin{pmatrix} Q_1^t \\ Q_2^t \end{pmatrix} = Q_1Q_1^t + Q\,Q_2^t \qquad (1.12)$$

$$I=Q^tQ;\ Q_1^tQ_1=I;\ Q_1^tQ_2=0;\ Q_2^tQ_1=0;\ Q_2^tQ_2=I$$

(c) $$\begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} Q_1^t \\ Q_2^t \end{pmatrix} \cdot A \equiv \begin{pmatrix} Q_1^t\,A=R \\ Q_2^t\,A=0 \end{pmatrix} \qquad (1.13)$$

(d) The minimal least square solution μ_L is found by solving

$$R\mu_L = Q_1^t g \qquad (1.14)$$

In fact, considering eqs. (1.2) and (1.9)-(1.4), and in a similar way to eq. (1.4), it results

$$b=g-A\mu_L=g-Q_1RR^{-1}Q_1^tg=g-Q_1Q_1^tg=$$
$$=(I-Q_1Q_1^t)g=Q_2Q_2^tg \qquad (1.15)$$

being

$$A\mu_L=Q_1Q_1^tg \qquad (1.16)$$

where μ is calculated by formula (1.14). For any other vector, say $\mu_L+\varepsilon$ (where ε is a vector) its norm $||b_1||^2$ is greater than $||b||^2$. In effect,

$$b_1=g-A(\mu_L+\varepsilon)=b-A\varepsilon \qquad (1.17)$$

$$||b_1||^2=||b-A\varepsilon||^2=||b||^2+||A\varepsilon||^2-2b^tA\varepsilon$$

$$= ||b||^2 + ||A\varepsilon||^2 - 2g^t Q_2 Q_2^t A\varepsilon$$
$$= ||b||^2 + ||A\varepsilon||^2 > ||b||^2 \qquad (1.18)$$

since $Q_2^t A = 0$, and being A a full rank matrix it results that $A\varepsilon \neq 0$ for any ε.

For the rest of the paper we will not use matrix Q_2. In order to simplify notation let substitute Q_1 by Q. Then the new notation is A=QR and it is termed QR-factorization of matrix A. Formula (1.14) is more stable than formula (1.8), since from (1.14) it results

$$\mu_{L_t} = Q_t^t \, g / R_{tt}$$

$$\mu_{L_i} = (Q_i^t g - \sum_{r=i+1}^{t} R_{ir} \mu_{L_r}) / R_{ii} \qquad (1.19)$$

for i=t-1, ...1

where Q_i is the i-th column of matrix Q. Then formula (1.19) is not losing precision if the computation of $Q^t g$ is correctly made. It is interesting to point out that in this computation it is implicit the orthogonality of matrix Q (in this case it means that $Q^t Q = I$, but $QQ^t \neq I$); if while obtaining matrix Q there are rounding errors, formula (1.19) is also unstable although it is better than formula (1.8).

Then the goal is to calculate the orthogonal matrix Q and the upper triangular matrix R (Sec.2), the minimum residual vector b (Sec. 3) and the unknown vector μ_L (Sec.4) with the maximum accuracy that it is possible in the computer. We will use the Modified Gram-Schmidt QR factorization. See e.g. /2 pp. 381-385/, /3 p. 313/, /4/ and /5/, although the technical details make a strong difference in the performance of any procedure.

2. CALCULATION OF MATRICES Q AND R

Matrix Q (dimension nxt) is calculated in t iterations in a recurrence way, so that at iteration 0 we assign $Q^{(0)} = A$ and at the end of iteration k, matrix $Q^{(k)}$ has the following expression

$$Q^{(k)} = (q_1^{(k)}, \ldots, q_k^{(k)}, \ldots, q_t^{(k)}) \qquad (2.1)$$

where q_i is the i-th column vector of matrix Q, so that vectors $q_1^{(k)}$ to $q_k^{(k)}$ take the definitive vectors $q_1 (\equiv q_1^{(t)})$ to $q_k (\equiv q_k^{(t)})$ of matrix $Q (\equiv Q^{(t)})$.

In succesive iterations (from k+1 to t) vectors $q_{k+1}^{(k)}$ to $q_t^{(t)}$ will be transformed in the definitive vectors $q_{k+1} (\equiv q_{k+1}^{(t)})$ to $q_t (\equiv q_t^{(t)})$ of matrix Q. Then at iteration k, vectors $q_1^{(k)}$ to $q_k^{(k)}$ are orthonormals, that is

$$q_i^{(k)t} \, q_j^{(k)} = \begin{cases} 1 & i=j \\ 0 & i \neq j \end{cases} \qquad (2.2)$$

for i,j = 1, 2, ..., k

Matrix R is calculated in a similar way, so that $R_{ij} = 0$ for i>j and i,j = 1, 2, ..., t. The row vector R_k of matrix R is obtained at the same iteration k where vector q_k is obtained. At iteration k the procedure is as follows.

Normalized Q algorithm (NQA):

(1) Obtain norms

$$N_i^{(k)} \equiv \left(\sum_{h=1}^{n} q_{hi}^{(k)2} \right)^{1/2} \qquad (2.3)$$

for i=k, k+1, ..., t

Note that the square root is not calculated in an exact way because of the limited representation of floating-point numbers in the computer. See below a slightly modification that avoids this possible inestability.

(2) Let $N_{k'}^{(k)}$ be the maximum of norms (2.3). If there are several norms with the same maximum value, we arbitrarily take the norm with the first subscript. Since we assume that A is a full rank matrix, $N_{k'}^{(k)} \neq 0$.

(3) If k'>k, the k'-th columns of matrix $Q^{(k)}$ are interchanged. Also we interchange the elements $R_{ik'}$ and R_{ik} of row vectors R_i (for i=1, 2, ..., k-1). After the interchange we will continue calling $q_i^{(k)}$ to the i-th column of matrix $Q^{(k)}$ and R_i to the i-th row of matrix R.

(4) Obtain vector q_k for the iteration k+1 (it will be the definitive vector q_k of matrix Q).

$$q_k^{(k+1)}=q_k^{(k)} \;/\; N_k^{(k)} \qquad (2.4)$$

(5) Obtain elements R_{ki} (i=k, k+1, ..., t) of row vector R_k

$$R_{kk}=q_k^{(k+1)^t} q_k^{(k)}=(q_k^{(k)^t} q_k^{(k)}) \;/\; N_k^{(k)}= N_k^{(k)} \qquad (2.5a)$$

$$R_{ki}=q_k^{(k+1)^t} q_i^{(k)}=(q_k^{(k)^t} q_i^{(k)})/N_k^{(k)} \qquad (2.5b)$$

for i=k+1, ..., t

(6) Calculate vectors q for iteration k+1

$$q_i^{(k+1)}=q_i^{(k)}-R_{ki}q_k^{(k+1)} \text{ for } i=k+1,\ldots,t \qquad (2.6)$$

The orthogonality (2.2) implies that column vectors $q_i^{(k)}$ (i=1, 2, ..., k) are theoretically normalized at the end of iteration k - (i.e. $\sum_{h=1}^{n} q_{hi}^{(k)^2}$ should be equal to 1), but eq. (2.4) is not very stable since the calculations of $N_i^{(k)}$ (2.3) are not exact. And - this inestability is augmented while calculating $q_i^{(k+1)}$ (2.6). In order to avoid it we modify the above algorithm mainly by not executing step (4) and storing in scalar d_k the norm $N_k^{(k)^2}$ that was calculated in steps (2)-(3); at the end of the new algorithm, the t-column vector d storages the norms of vectors q_1, q_2, ..., q_t that are no longer normalized. Then the expression of the orthogonality of matrix Q will be

$$(q_i^t\, q_j)/(N_i N_j)=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases} \qquad (2.7a)$$

for i, j = 1, 2, ..., t; that is

$$Q^t Q \delta^t \delta = I \qquad (2.7b)$$

where the i-th element of the t-column vector δ is $1/d_i^{1/2}$.

Non-Normalized Q algorithm (NNQA):

At iteration k of the new algorithm:

(1) Obtain elements d_i (i=k, k+1, ..., t) of vector d

$$d_i= N_i^{(k)^2} \equiv \sum_{h=1}^{n} q_{hi}^{(k)^2} \qquad (2.8)$$

(2) -(3) as in NQA

(4) Assign directly

$$q_k^{(k+1)}=q_k^{(k)} \qquad (2.9)$$

(5) As in NQA, but normalizing the row vector R_k with $N_k^{(k)}$. See eq. (2.12).

Then,

$$R_{kk}=((q_k^{(k)^t} q_k^{(k)})/N_k^{(k)})/N_k^{(k)}= 1 \qquad (2.10a)$$

$$R_{ki}=((q_k^{(k)^t} q_i^{(k)})/N_k^{(k)})/N_k^{(k)}=$$

$$=(q_k^{(k)^t} q_i^{(k)})/d_k \equiv (q_k^{(k+1)^t} q_i^{(k)})/d_k \qquad (2.10b)$$

(6) Obtain vectors q for iteration k+1. We use formula (2.6), but considering that vector $q_k^{(k+1)}$ is not normalized. Note that now element R_{ki} is normalized in formula (2.10b) Then, we have the same result

$$q_i^{(k+1)}=q_i^{(k)}-R_{ki}N_k^{(k)}(q_k^{(k+1)}/N_k^{(k)})=$$

$$q_i^{(k)}-R_{ki}q_k^{(k+1)} \qquad (2.11)$$

for i=k+1, ..., t. Note that since NNQA is not using $N_k^{(k)}$, but d_k (and it is used only at the last calculations), there are fewer rounding errors. Although in NNQA we calculate Q and R without normalizing the columns of matrix Q and normalizing the rows of matrix R, the QR factorization is the same. In fact, we have reduced the rounding errors by affecting the norm of each column only once to the whole column vector (see eq. (2.10)). If we consider that Q and R are the matrices calculated by NNQA, and Q' and R' are the matrices calculated by NQA, theoretically - Q'R'=QR. In effect, element A_{ij} of matrix A can be written

$$A_{ij}=\sum_{h=1}^{j} Q'_{ih}\, R'_{hj}=\sum_{h=1}^{j} (Q_{ih}/N_h)(R_{hj}N_h)=$$

$$=\sum_{=1}^{j} Q_i\, R_j \,. \qquad (2.12)$$

where $R_{jj}=1$ (eq. (2.10a)). Then A=QR=Q'R'. We may see that when Q is a non-normalized matrix and R is a normalized matrix with the identity diagonal, the procedure is more stable than when Q is normalized and R is non-normalized.

3. CALCULATION OF RESIDUAL VECTOR B=G-AU

Since A=QR (being Q and R calculated by NNQA)

it is possible to obtain μ with formula -- (1.19). But it is based on the orthogonality of matrix Q ($A\mu=g$; $A\mu=QQ^tg$; $Q^tA\mu=Q^tQQ^tg$; $R\mu=Q^tg$); then if while calculating matrix Q there are some rounding errors, it is possible that Q is not orthogonal and μ may be unstable. Alternatively, it is better the algorithm described below. The main idea is to avoid any possibility of rounding errors by substituting any intermediate operation -- $q_i^tq_j$ for $i\neq j$ by zero (where zero is its theoretical value). Firstly we obtain the residual b and second the unknown vector μ is obtained. The n-column vector b is obtained in t iterations (recall that Q is a nxt matrix) in recurrence way, so that at iteration 0 we assign $b^{(0)}=g$, and at the end of iteration t vector $b^{(t)}$ takes the residual vector b.

The justification of the procedure is as follows. Consider eqs. (1.15) and (1.16); note that Q' and R' is the notation of the matrices calculated by NQA and Q and R is the notation of matrices calculated by NNQA. Eqs. (1.15) and (1.16) assume the orthogonality $Q'^tQ'=I$ (or $Q^tQ\delta^t\delta=I$). Then

$$b=g-Q'Q'^tg=g-\sum_{i=1}^{t}(q_i'^tg)q_i'$$
$$=g-\sum_{i=1}^{t-1}((q_i^tg)/d_i)q_i -$$
$$-((q_t^tg)/d_t)q_t \qquad (3.1)$$

Defining

$$b^{(t-1)}=g-\sum_{i=1}^{t-1}((q_i^tg)/d_i)q_i \qquad (3.2)$$

results

$$b=b^{(t-1)}-((q_t^tg)/d_t)q_t \qquad (3.3)$$

Similarly we may substitute eq. (3.2) by

$$b^{(k)}=b^{(k-1)}-((q_k^tg)/d_k)q_k \qquad (3.4a)$$

for k=1, 2, ..., t

where by definition

$$b^{(k-1)}\equiv g-\sum_{i=1}^{k-1}((q_i^tg)/d_i)q_i \qquad (3.4b)$$

where $b^{(k)}$ for k=t is the residual vector b (3.3).

The stability of procedure (3.4) depends on the accuracy of matrix Q calculated by NNQA. In effect, if we premultiply $q^t{}_k/N_k$ by eq. (3.4b), it results

$$(q_k^tb^{(k-1)})/N_k=(q_k^tg)/N_k-$$
$$(q_k^t/N_k)\sum_{i=1}^{k-1}((q_i^tg)/N_i)(q_i/N_i)$$
$$=(q_k^tg)/N_k-\sum_{i=1}^{k-1}((q_i^tg)/N_i)((q_k^tq_i)/(N_kN_i)) \qquad (3.5a)$$

The calculation $(q_k^tg)/d_k$ (where $d_k\equiv N_k^2$) of formula (3.4a) will be as precise as it is eq. (3.5a); if in this eq., $q_k^tq_i\neq0$ (i=1, 2, ..., k-1 , then $i\neq k$), (q_k^tg) will not be -- exact and $b^{(k)}$ calculated by formula (3.4a) will be unstable. Since we know that without rounding errors, $q_k^tq_i=0$, it is substituted by zero and eq. (3.5a) gives

$$(q_k^tb^{(k-1)})/N_k=(q_k^tg)/N_k \therefore$$

$$q_k^tb^{(k-1)}=q_k^tg \qquad (3.5b)$$

Then the final formula to calculate $b^{(k)}$ is

$$b^{(k)}=b^{(k-1)}-((q_k^tb^{(k-1)})/d_k)q_k \qquad (3.6)$$

for k=1, 2, ..., t; and $b^{(0)}=g$, $\quad b\equiv b^{(t)}$

4. CALCULATION OF THE UNKNOWN VECTOR μ

With the same notation Q', R', Q and R used in the previous sections, the vector μ that minimizes norm $||b||^2$ (1.2) is found by solving eqs. (1.14): $R'\mu=Q'^tg$, that for element μ_k gives

$$R'_{kk}\mu_k+\sum_{i=k+1}^{t}R'_{ki}\mu_i=q_k'^tg \qquad (4.1a)$$

for k = 1, 2, ..., t-1

$$R'_{tt}\mu_t=q_t'^tg \qquad (4.1b)$$

By using matrices Q and R calculated by NNQA (see Sec. 2), eqs. (4.1) give

$$N_k\mu_k=(q_k^tg)/N_k-\sum_{i=k+1}^{t}N_kR_{ki}\mu_i \qquad (4.2a)$$

for k=1, 2, ..., t-1

$$N_t\mu_t=(q_t^tg)/N_t \qquad (4.2b)$$

Since $q_k^tg=q_k^tb^{(k-1)}$ (eq. 3.5b), eqs. (4.2) give

$$\mu_t=(q_t^t b^{(t-1)})/d_t \qquad (4.3a)$$

$$\mu_k=(q_k^t b^{(k-1)})/d_k-\sum_{i=k+1}^{t} R_{ki}\mu_i \qquad (4.3b)$$

for k=1, 2, ..., t-1; where $d_k \equiv N_k^2$.

We may note that formulae (3.6) and (4.3) - use the same intermediate operations; then we may avoid to repeat these calculations. Also note that square root calculations are avoided (this operation produces the most of the rounding errors in alternate procedures).

5. CONCLUSION

A procedure for solving an overdetermined - system of linear equations is revisited and some new technical details are introduced. - It calculates the orthogonal matrix Q (NNQA in Sec. 2), the residual vector b (3.6) and the unknown vector μ (4.6) with the maximum accuracy. It is much more stable than procedures that use the pseudoinverse of the system matrix.

6. ACKNOWLEDGEMENT

I thank Dr. Augustin Drubrulle and one of the referees for their helpful suggestions.

7. REFERENCES

/1/ ESCUDERO, L.F., "Lagrange multipliers estimates for constrained optimization", IBM Scientific Center Report, 1980, Palo Alto, California.

/2/ GOLUB, G.H., "Matrix decomposition and - statistical calculations", Academic Press London, 1969, pp. 381-385.

/3/ PETERS, G. and WILKINSON, J.H., "The -- least-square problem and pseudoinverses", Computer Journal 13, 1970, pp. 309-316.

/4/ RICE, J.R., "Experiments on Gram-Schmidt orthogonalization", Mathematical Computations 20, 1966, pp. 325-328.

/5/ WAMPLER, R.H., "Solutions to weighted least squares problems by Modified Gram-Schmidt with iterative refinement, ACM - Transactions on Mathematical Software 5, -- 1979, pp. 457-465.