

**Dynamic Programming and Pseudo-inverses**

*Y. Fan*

*School of Engineering, University of Southern California*

*R. Kalaba*

*Department of Economics, University of Southern California*

**Summary**

Dynamic programming can be used to obtain the generalized inverse of a matrix.

**Key words:** Generalized Inverse, Dynamic Programming, Greville's Algorithm, Least Squares.

## 1. INTRODUCTION

Generalized inverses of matrixes play a role in least squares problems. If we wish to minimize  $q = (Ax-b)^T(Ax-b)$ , a solution is given by  $x = A^+b$ , where  $A^+$  is the Moore-Penrose inverse of  $A$ . Furthermore, if the solution is not unique, then  $A^+b$  is the solution of minimal length, which is unique. But generalized inverses occur in many other ways. They are crucial in the theory of linear associative memories [1], in regression analysis [2], and the theory of constrained mechanical motion [3].

There are many approaches available for the determination of generalized inverses. Among these are the methods of Decell and Greville [4,5]. A prominent role is played by the resolution of a matrix into its singular value decomposition [3]. But the functional equation approach of dynamic programming is also available, because we may consider a matrix as being built up one column at a time. There is a long history of applications of dynamic programming to least squares problems [6]. But this is the first time, to our knowledge, that it has been used to obtain the generalized inverse of a matrix. The main purpose of this paper is to show how this is done and to elucidate the connection between dynamic programming and the Greville approach [5], also a sequential method.

In the next section we present the approach of determination of generalized inverses of matrices using Bellman's principle of optimality. This results in an algorithm, which may be called the  $\alpha$ Q $\beta$ R procedure. Then Greville's algorithm is reviewed briefly. Next it is shown that the  $\alpha$ Q $\beta$ R algorithm produces enough information to determine the generalized inverse of a matrix, whether it is of full rank or not. Finally, some results of numerical experiments are presented, and conclusions are given in a discussion. Computer programs in MATLAB are provided in an appendix.

## 2. DERIVATION OF $\alpha$ Q $\beta$ R ALGORITHM

This dynamic program for calculating generalized inverse is inspired by the  $\alpha$ Q $\beta$ R algorithm which seeks for the vector  $x$  through a dynamic program such that  $|Ax - b|^2$  is a minimum, and the length of  $x$  is as small as possible. Here,  $A$  is an  $m$  by  $n$  matrix,  $b$  is a column vector of dimension  $m$ , and  $x$  is a column vector of dimension  $n$ .

Let us first briefly look at how the  $\alpha$ Q $\beta$ R algorithm solves least square problems. Two cost functions are introduced in this algorithm. The first one is

$$f_k(b) = \text{the smallest square of the length of the vector } A_k x^k - b, \quad (2.1)$$

where  $A_k$  is a matrix consisting of the first  $k$  columns of  $A$ , and  $x^k$  is a column vector of dimension  $k$ . The second one is

$$g_k(b) = \text{the smallest square of the length of the vector } x^k, \quad (2.2)$$

where  $x^k$  is subject to the restriction

$$|A_k x^k - b|^2 = \min. \quad (2.3)$$

Individual columns of  $A$  are denoted by  $a_1, a_2, \dots, a_n$ . Suppose that  $f_{k-1}(b)$  and  $g_{k-1}(b)$  are known. Then, the functions  $f_k(b)$  and  $g_k(b)$  are obtained using the principle of optimality [6]. The dynamic programming procedure for updating  $f_k(b)$  and  $g_k(b)$  based on knowledge of the functions  $f_{k-1}(b)$  and  $g_{k-1}(b)$  depends on whether  $a_k$  is linearly dependent on  $a_1, a_2, \dots, a_{k-1}$ . If  $a_k$  is linearly dependent on  $a_1, a_2, \dots, a_{k-1}$ , then

$$f_k(b) = f_{k-1}(b), \quad (2.4)$$

because a linear combination of  $a_1, a_2, \dots, a_{k-1}, a_k$  can approximate the vector  $b$  no better than a linear combination of  $a_1, a_2, \dots, a_{k-1}$ . Also,

$$g_k(b) = \min_{x_k} [x_k^2 + g_{k-1}(b - x_k a_k)], \quad (2.5)$$

where  $x_k$  is a scalar, the  $k^{th}$  component of vector  $x^k$ . The reason is that once  $x_k$  is chosen, then the immediate penalty is  $x_k^2$ . The remaining components,  $x_1, x_2, \dots, x_{k-1}$ , have to be chosen to bring a linear combination of  $a_1, a_2, \dots, a_{k-1}$  as close as possible to the new target vector, which is  $b - x_k a_k$ . Furthermore, the sum of  $x_1^2 + x_2^2 + \dots + x_{k-1}^2$  must be a minimum.

In the case that  $a_k$  is linearly independent of  $a_1, a_2, \dots, a_{k-1}$ , with any choice of the scalar  $x_k$  we must approximate the new target vector  $b - x_k a_k$  as closely as possible by the sum  $x_1 a_1 + \dots + x_{k-1} a_{k-1}$ . Thus

$$f_k(b) = \min_{x_k} f_{k-1}(b - x_k a_k). \tag{2.6}$$

If the minimizing value of the scalar  $x_k$  is  $x_k^*$ , then

$$g_k(b) = (x_k^*)^2 + g_{k-1}(b - x_k^* a_k). \tag{2.7}$$

Now, let us look at the function

$f_k(b)$  = the square of the length of the residual  $(A_k x^k - b)$ , when the vector  $x^k$  is chosen optimally, where  $\dim b = m$ , and  $A_k$  consists of the first  $k$  columns of  $A$ .

When  $k = 1$  we have

$$f_1(b) = \min (a_1 x_1 - b)^T (a_1 x_1 - b). \tag{2.8}$$

The minimizing condition is

$$a_1^T a_1 x_1 - a_1^T b = 0, \tag{2.9}$$

so that

$$x_1^* = \frac{a_1^T b}{a_1^T a_1} = a_1^+ b. \tag{2.10}$$

where  $a_1^+$  is the generalized inverse of the vector  $a_1$  (assuming that  $a_1 \neq 0$ ). It follows that

$$f_1(b) = b^T [I - a_1 a_1^+] b. \tag{2.11}$$

The function  $f_1(b)$  is a quadratic form in  $b$  which can be written as

$$f_1(b) = b^T Q_1 b, \quad (2.13)$$

where  $Q_1$  is a symmetric positive semi-definite  $m$  by  $m$  matrix.

To prove that  $f_k(b)$  is a quadratic form in  $b$ , that is

$$f_k(b) = b^T Q_k b, \quad (2.15)$$

where  $Q_k$  is a symmetric positive-definite  $m$  by  $m$  matrix, we look at two cases separately, depending on whether the vector  $a_k$  is linearly dependent on the previous vectors or not.

Assume first that  $a_k$  is independent of  $a_1, a_2, \dots, a_{k-1}$ . The  $k^{\text{th}}$  element of vector  $x$  is denoted as  $s$ . From Eq. (2.6), we have

$$f_k(b) = \min_s f_{k-1}(b - a_k s), \quad k = 2, 3, \dots, n. \quad (2.16)$$

Assume that

$$f_{k-1}(b) = b^T Q_{k-1} b. \quad (2.17)$$

Then we have

$$f_k(b) = \min_s \{(b - sa_k)^T Q_{k-1} (b - sa_k)\}. \quad (2.18)$$

The function to be optimized is

$$\begin{aligned} & \{(b - sa_k)^T Q_{k-1} (b - sa_k)\} \\ &= (b^T Q_{k-1} - sa_k^T Q_{k-1})(b - sa_k), \\ &= b^T Q_{k-1} b - sa_k^T Q_{k-1} b - b^T Q_{k-1} sa_k + s^2 a_k^T Q_{k-1} a_k. \end{aligned} \quad (2.19)$$

To obtain the optimal solution, we let  $\frac{d\{\}}{ds} = 0$ , so that

$$\frac{d\{\}}{ds} = 2sa_k^T Q_{k-1} a_k - a_k^T Q_{k-1} b - b^T Q_{k-1} a_k = 0. \quad (2.20)$$

Note that since  $a_k^T Q_{k-1} b$  is a scalar,  $a_k^T Q_{k-1} b = (a_k^T Q_{k-1} b)^T = b^T Q_{k-1} a_k$ . Therefore, the optimal solution is

$$s^* = \frac{b^T Q_{k-1} a_k}{a_k^T Q_{k-1} a_k} = \frac{a_k^T Q_{k-1} b}{a_k^T Q_{k-1} a_k}. \quad (2.21)$$

Apply the two forms of the optimal solution  $s$ , and denote  $\alpha_k = Q_{k-1} a_k$ . Eq. (2.19) can be rewritten as

$$f_k(b) = b^T Q_{k-1} b - 2b^T \alpha_k \frac{\alpha_k^T b}{a_k^T \alpha_k} + \frac{b^T \alpha_k}{a_k^T \alpha_k} (a_k^T \alpha_k) \frac{\alpha_k^T b}{a_k^T \alpha_k}. \quad (2.22)$$

Therefore, we have

$$f_k(b) = b^T \left( Q_{k-1} - \frac{\alpha_k \alpha_k^T}{a_k^T \alpha_k} \right) b = b^T Q_k b. \quad (2.23)$$

The recursive updating of  $Q_{k-1}$  is

$$Q_k = Q_{k-1} - \frac{\alpha_k \alpha_k^T}{a_k^T \alpha_k}. \quad (2.24)$$

Next assume  $a_k$  is dependent upon  $a_1, a_2, \dots, a_{k-1}$ . Then  $f_k(b) = f_{k-1}(b)$ , because a linear combination of  $a_1, a_2, \dots, a_{k-1}, a_k$  cannot be brought closer to  $b$  than a linear combination of  $a_1, a_2, \dots, a_{k-1}$ . Thus we have  $Q_k = Q_{k-1}$ .

Now, let us look at the second function

$g_k(b) =$  the smallest square of the length of the vector  $x^k$ , for which  $|A_k x^k - b|$  is a minimum.

When  $k = 1$  we know, from the previous discussion, that in the one stage process

$$x_1^* = \frac{a_1^T b}{a_1^T a_1} = a_1^+ b, \quad (2.25)$$

where  $a_1^+$  is the generalized inverse of the vector  $a_1$  (assuming that  $a_1 \neq 0$ ). It follows that

$$g_1(b) = b^T (a_1^+)^T a_1^+ b. \quad (2.26)$$

The functions  $g_1(b)$  is a quadratic form in  $b$  which can be written as

$$g_1(b) = b^T R_1 b, \quad (2.27)$$

where  $R_l$  is a symmetric positive semi-definite  $m$  by  $m$  matrix.

To prove that  $g_k(b)$  is also a quadratic form in  $b$ , that is

$$g_k(b) = b^T R_k b, \quad (2.28)$$

we look at two cases separately, depending on whether the vector  $a_k$  is linearly dependent on the previous vectors or not.

Assume that  $a_k$  is independent of  $a_1, a_2, \dots, a_{k-1}$ . From eq. (2.21) we know that

$$s^* = \frac{\alpha_k^T b}{\alpha_k^T a_k}. \quad (2.29)$$

Apply  $s^*$  to eq. (2.7), which is

$$g_k(b) = (s^*)^2 + g_{k-1}(b - s^* a_k).$$

Then we have

$$\begin{aligned} g_k(b) &= (s^*)^2 + (b - s^* a_k)^T R_{k-1} (b - s^* a_k), \\ &= \frac{b^T \alpha_k}{\alpha_k^T a_k} \frac{\alpha_k^T b}{\alpha_k^T a_k} + b^T \left[ I - \frac{\alpha_k a_k^T}{\alpha_k^T a_k} \right] R_{k-1} \left[ I - \frac{a_k \alpha_k^T}{\alpha_k^T a_k} \right] b, \\ &= b^T \left\{ \left[ I - \frac{\alpha_k a_k^T}{\alpha_k^T a_k} \right] R_{k-1} \left[ I - \frac{a_k \alpha_k^T}{\alpha_k^T a_k} \right] + \frac{\alpha_k \alpha_k^T}{(\alpha_k^T a_k)^2} \right\} b. \end{aligned} \quad (2.30)$$

Assume next that  $a_k$  is dependent upon  $a_1, a_2, \dots, a_{k-1}$ . From the previous part we know that the solution to Eq. (2.3) is not unique, one of which would result in the minimum length of the vector  $x^k$ . The  $k^{\text{th}}$  element of solution  $x$  is denoted as  $s$ ; the recursive relationship is shown as

$$g_k(b) = \min_s \{ s^2 + g_{k-1}(b - s a_k) \}. \quad (2.31)$$

Assuming that

$$g_{k-1}(b) = b^T R_{k-1} b, \quad (2.32)$$

we have

$$g_k(b) = \min_s \{s^2 + (b - sa_k)^T R_{k-1} (b - sa_k)\}, \quad (2.33)$$

$$\begin{aligned} \{\dots\} &= s^2 + (b^T - sa_k^T) R_{k-1} (b - sa_k), \\ &= s^2 + (b^T R_{k-1} - sa_k^T R_{k-1}) (b - sa_k), \\ &= s^2 + b^T R_{k-1} b - 2sa_k^T R_{k-1} b + s^2 a_k^T R_{k-1} a_k. \end{aligned} \quad (2.34)$$

Denote  $R_{k-1} a_k$  as  $\beta_k$ , and let  $\frac{d\{\}}{ds} = 0$ . We have

$$2s - 2\beta_k^T b + 2sa_k^T \beta_k = 0. \quad (2.35)$$

Therefore, the optimal solution is

$$s^* = \frac{b^T \beta_k}{1 + a_k^T \beta_k}. \quad (2.36a)$$

Since  $s$  is a scalar, we also have

$$s^* = \frac{\beta_k^T b}{1 + a_k^T \beta_k}. \quad (2.36b)$$

Apply Eq. (2.36a) and Eq. (2.36b) to the function  $g^k(b)$ . It is seen that

$$\begin{aligned} g_k(b) &= \frac{b^T \beta_k}{1 + a_k^T \beta_k} (1 + a_k^T \beta_k) \frac{\beta_k^T b}{1 + a_k^T \beta_k} + b^T R_{k-1} b - 2 \frac{b^T \beta_k}{1 + a_k^T \beta_k} \beta_k^T b, \\ &= b^T \left( R_{k-1} - \frac{\beta_k \beta_k^T}{1 + a_k^T \beta_k} \right) b. \end{aligned} \quad (2.37)$$

Therefore, it has been proved that  $g_k(b)$  is also a quadratic form in  $b$ ; that is,

$$g_k(b) = b^T R_k b, \quad (2.38)$$

where  $R_k = R_{k-1} - \frac{\beta_k \beta_k^T}{1 + a_k^T \beta_k}$ .

In the above procedure, one needs to determine whether  $a_k$  is linearly dependent on  $a_1, a_2, \dots, a_{k-1}$ , or not. It is to be proved that

$$a_k = \alpha_k + \text{lin. com. of } a_1, a_2, \dots, a_{k-1}, \quad (2.39)$$

where  $\alpha_k = Q_{k-1}a_k$ . Also,  $\alpha_k$  is to be proved orthogonal to  $a_1, a_2, \dots, a_{k-1}$ . See the Appendix for details. Therefore, instead of directly testing whether  $a_k$  is linearly dependent on  $a_1, a_2, \dots, a_{k-1}$ , we can examine whether  $\alpha_k$  is the null vector or not. That is, if

$$\alpha_k = Q_{k-1}a_k = 0,$$

then  $a_k$  is linearly dependent on  $a_1, a_2, \dots, a_{k-1}$ ; otherwise, if

$$\alpha_k = Q_{k-1}a_k \neq 0,$$

then  $a_k$  is linearly independent of  $a_1, a_2, \dots, a_{k-1}$ .

It will be proved in Section 4 that all the information needed to get generalized inverses has been obtained in the above procedure. The steps are as below:

$$\alpha_1 = a_1, \quad (2.40)$$

$$Q_1 = I - \frac{\alpha_1 \alpha_1^T}{\alpha_1^T \alpha_1} = I - a_1 \alpha_1^+, \quad (2.41)$$

$$R_1 = (\alpha_1^+)^T a_1^+. \quad (2.42)$$

Then for each value of  $k, k = 2, 3, \dots, n$ , there are two cases. If

$$\alpha_k = Q_{k-1}a_k = 0, \quad (2.43)$$

then

$$Q_k = Q_{k-1}, \quad (2.44)$$

$$\beta_k = R_{k-1}a_k, \quad (2.45)$$

$$R_k = R_{k-1} - \frac{\beta_k \beta_k^T}{1 + a_k^T \beta_k}. \quad (2.46)$$

If, on the other hand,

$$\alpha_k = Q_{k-1} a_k \neq 0, \quad (2.47)$$

then

$$\alpha_k^+ = \frac{\alpha_k^T}{\alpha_k^T \alpha_k}, \quad (2.48)$$

$$Q_k = Q_{k-1} - \alpha_k \alpha_k^+, \quad (2.49)$$

$$R_k = (\alpha_k^+)^T \alpha_k^+ + (I - \alpha_k \alpha_k^+)^T R_{k-1} (I - \alpha_k \alpha_k^+). \quad (2.50)$$

### 3. DERIVATION OF GREVILLE'S ALGORITHM

Because of its extensive applicability, Greville's result is widely stated, but no constructive proof is provided because of the perceived complexity of his solution technique. But a simple derivation has been given by Udvardia and Kalaba (1997) [5]. The derivation is as below.

Suppose the MP inverse of the matrix  $A$ , denoted as  $A^+$ , is known. We want to get the solution to the least square problem  $Bx \cong b$ . The matrix  $B$  is partitioned as  $[A \ a]$ , and the vector  $x$  is partitioned as  $\begin{pmatrix} z \\ s \end{pmatrix}$ , where  $z$  is a vector of dimension  $k-1$ , and  $s$  is a scalar. We

therefore rewrite  $Bx \cong b$  as  $(A \ a) \begin{pmatrix} z \\ s \end{pmatrix} \cong b$ . Then

$$Az + as \cong b, \quad (3.1)$$

$$Az \cong b - as. \quad (3.2)$$

With the knowledge of the generalized inverse, we know that the solution is

$$z = A^+ (b - as). \quad (3.3)$$

Substituting Eq. (3.3) into Eq. (3.2), we have

$$AA^+(b - as) \cong b - as, \quad (3.4)$$

$$(I - AA^+)as \cong (I - AA^+)b. \quad (3.5)$$

Denote  $(I - AA^+)a$  as  $c$ . If  $c \neq 0$ , we have

$$s = \frac{a^T(I - AA^+)}{a^T(I - AA^+)a} (I - AA^+)b. \quad (3.6)$$

Since  $(I - AA^+)$  is symmetric and idempotent, which can be proved by using the properties of generalized inverses, we have

$$s = \frac{a^T(I - AA^+)}{a^T(I - AA^+)a} b, \quad (3.7)$$

$$s = c^+b. \quad (3.8)$$

On the other hand, we know the solution of  $(A \ a) \begin{pmatrix} z \\ s \end{pmatrix} \cong b$  is

$$\begin{pmatrix} z \\ s \end{pmatrix} = (A \ a)^+ b. \quad (3.9)$$

Recalling the solution  $z$  and  $s$  obtained in the previous part, we have,

$$(A \ a)^+ b = \begin{pmatrix} A^+ - A^+ac^+ \\ c^+ \end{pmatrix} b. \quad (3.10)$$

Since the vector  $b$  is arbitrary,

$$(A \ a)^+ = \begin{pmatrix} A^+ - A^+ac^+ \\ c^+ \end{pmatrix}, \text{ if } c \neq 0. \quad (3.11)$$

Now let us consider the case when  $c = 0$ ; namely,  $c$  is an  $m$ -dimensional null vector.

From Eq. (3.5), we know that when  $c = 0$ ,  $s$  can be any scalar, which means the solution vector

is not unique. Instead of picking any solution that holds  $(A \ a) \begin{pmatrix} z \\ s \end{pmatrix} \cong b$ , we want a solution with the minimum length, which means

$$z^T z + s^2 = \min . \tag{3.12}$$

Denote  $(A^+b)$  as  $u$  and  $(A^+a)$  as  $v$ . We have

$$z = A^+(b - as) = u - vs, \tag{3.13}$$

Therefore

$$(u - vs)^T (u - vs) + s^2 = \min , \tag{3.14}$$

$$u^T u - 2sv^T u + v^T vs^2 + s^2 = \min . \tag{3.15}$$

The value of  $s$  that minimizes the length is obtained from

$$(1 + v^T v)s = v^T u = v^T A^+ b, \tag{3.16}$$

yielding

$$s = \frac{v^T A^+}{1 + v^T v} b, \tag{3.17}$$

so that

$$(A \ a)^+ = \begin{pmatrix} A^+ - A^+ a \frac{v^T A^+}{1 - v^T v} \\ \frac{v^T A^+}{1 - v^T v} \end{pmatrix} \text{ if } c = 0. \tag{3.18}$$

Eq. (3.11) and Eq. (3.18) constitute the updating procedure.

#### 4. GENERALIZED INVERSES AND $\alpha$ QBR

Recall the fact that the solution of the minimal norm least squares problem  $(Ax - b)^T (Ax - b) = \min$  can also be obtained by  $x = A^+ b$ , where  $A^+$  is the pseudo-inverse of  $A$ . It is natural to seek  $A^+$  through the  $\alpha$ QR algorithm.

Greville's algorithm shows how to pass from knowledge of  $A_{k-1}^+$ , the pseudo-inverse of  $A_{k-1}$ , which is the first  $k-1$  columns of matrix  $A$ , to the pseudo-inverse  $A_k^+$  of the matrix  $A_k$ , which is the first  $k$  columns of matrix  $A$ . The vector  $c$  plays a crucial role in the updating. Let us consider what the vector  $c$  actually is. Assume the matrix  $A$  contains  $n$  columns, and column vector  $w$  contains  $n$  scalars  $w_1, w_2, \dots, w_n$ . They are denoted as

$$A = (a_1 \quad a_2 \quad \dots \quad a_n), \quad w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}.$$

Therefore,  $Aw$  is a linear combination of the columns in matrix  $A$ . Since  $c = (I - AA^+)a$ ,

$$c^T Aw = a^T (I - AA^+)^T Aw. \tag{4.1}$$

Because  $I - AA^+$  is symmetric,

$$c^T Aw = a^T (I - AA^+)Aw, \tag{4.2a}$$

$$= a^T (Aw - AA^+Aw). \tag{4.2b}$$

Recall one of the properties of the MP generalized inverse,  $AA^+A = A$ . We conclude that

$$c^T Aw = 0, \tag{4.3}$$

which means that vector  $c$  is orthogonal to any linear combination of the columns of matrix  $A$ .

Moreover, from  $c = (I - AA^+)a$  we have

$$a = c + AA^+a, \tag{4.4}$$

where  $A^+a$  can be considered as some vector  $w$ . Therefore,  $a$  is a linear combination of the previous columns in matrix  $A$  plus a vector  $c$  which is orthogonal to all those columns in matrix  $A$ . Therefore, the new column,  $a$ , is determined as independent of all previous columns, when  $c$  is not the null vector.

Now we consider a vector  $\alpha_k$  in the  $\alpha Q\beta R$  algorithm. It can be shown that the vector  $\alpha_k$  is orthogonal to the vectors  $a_1, a_2, \dots, a_{k-1}$ . Instead of applying a vector  $c$  as Greville does, the updating of  $A_{k-1}^+$  to  $A_k^+$  can be accomplished in terms of either  $\alpha_k$  or  $\beta_k$ , depending on whether  $\alpha_k \neq 0$  or  $\alpha_k = 0$ .

First let us consider the case in which  $c \neq 0$ . This means that the vector  $a_k$  has a non-null component that is orthogonal to the vector  $a_1, a_2, \dots, a_{k-1}$ . Consequently, this is the case in which  $\alpha_k$  is not linearly dependent upon  $a_1, a_2, \dots, a_{k-1}$ . The Greville updating is given by

$$A_k^+ = \begin{pmatrix} A_{k-1}^+ - A_{k-1}^+ a_k c^+ \\ c^+ \end{pmatrix}, \quad (4.5)$$

which requires a knowledge of the vector  $c$ , in addition to  $A_{k-1}^+$  and  $a_k$ . But the  $\alpha Q\beta R$  algorithm provides the vector  $\alpha_k$ , which is the component of  $a_k$  that is orthogonal to  $a_1, a_2, \dots, a_{k-1}$ , as we have seen earlier. Thus when  $A_{k-1}^+, a_k$  and  $\alpha_k$  are known, we may determine  $A_k^+$  as

$$A_k^+ = \begin{pmatrix} A_{k-1}^+ - A_{k-1}^+ a_k \alpha_k^+ \\ \alpha_k^+ \end{pmatrix}. \quad (4.6)$$

The second case is that in which  $c = 0$ . In this case  $a_k = A_{k-1} A_{k-1}^+ a_k$ , so that the vector  $a_k$  is linearly dependent on the columns of the matrix  $A_{k-1}$ . Greville's updating is given by the formula

$$A_k^+ = \begin{pmatrix} A_{k-1}^+ - A_{k-1}^+ a_k v^T \\ v^T \end{pmatrix}, \quad (4.7)$$

where

$$v = \frac{(A_{k-1}^+)^T A_{k-1}^+ a_k}{1 + a_k^T (A_{k-1}^+)^T A_{k-1}^+ a_k}. \quad (4.8)$$

To interpret these relations from the point of view of the  $\alpha$ Q $\beta$ R algorithm, we recall Eqs. (2.2) and (2.16):

$$\begin{aligned} g_k(b) &= \text{the smallest square of the length of the vector } x_k \text{ which minimizes } |A_k x_k - b|^2, \\ &= b^T R_k b. \end{aligned}$$

We know the solution is

$$x_k = A_k^+ b, \quad (4.9)$$

so that we may write

$$g_k(b) = b^T (A_k^+)^T A_k^+ b. \quad (4.10)$$

Thus, we see that

$$R_k = (A_k^+)^T A_k^+, \quad (4.11)$$

which should be consistent with the  $R_k$  obtained in the previous Eqs. (2.23) and (2.27).

This enables us to write

$$v = \frac{R_{k-1} a_k}{1 + a_k^T R_{k-1} a_k}, \quad (4.12)$$

$$v = \frac{\beta_k}{1 + a_k^T \beta_k}, \quad (4.13)$$

which expresses the vector  $v$  in terms of  $\beta_k$ .

These relations show how the updating of  $A_{k-1}^+$  to  $A_k^+$  is accomplished in terms of either  $\alpha_k$  or  $\beta_k$ , depending on whether  $\alpha_k = 0$  or  $\alpha_k \neq 0$ .

In addition, we know the solution to the problem  $|A_k x^k - b|^2 = \min$  is  $x^k = A_k^+ b$ . Thus

$$f_k(b) = (A_k A_k^+ b - b)^T (A_k A_k^+ b - b), \quad (4.14a)$$

$$= b^T (A_k A_k^+ - I)(A_k A_k^+ - I)b, \quad (4.14b)$$

$$= b^T (A_k A_k^+ - A_k A_k^+ - A_k A_k^+ + I)b, \quad (4.14c)$$

$$= b^T (I - A_k A_k^+)b. \quad (4.14d)$$

This shows that

$$Q_k = I - A_k A_k^+, \quad (4.15)$$

where  $Q_k$  is calculated during the  $\alpha Q\beta R$  algorithm,  $A_k$  is the first  $k$  columns of matrix  $A$ , and  $A_k^+$  is the generalized inverse of  $A_k$ .

## 5. NUMERICAL EXAMPLES

We have experimented with determining pseudo-inverses of matrix  $A$  by the  $\alpha Q\beta R$  updating. Due to computing resource limitations, we could not attempt to solve large problems. Two smaller examples are provided here to illustrate the applicability and procedure of obtaining pseudo inverses by dynamic programming. Because of the existence of round-off error, a very small number, tolerance, is introduced when implementing the algorithm. Any vector whose length is less than the tolerance is treated as a null vector. Example 1 contains detailed steps, while in the second example we have omitted details of solution to conserve space.

### Example 1

Let us look at an example where matrix  $A$  is a 5 by 3 matrix and is of rank 2.

$$A = \begin{bmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 + \varepsilon \end{bmatrix}, \text{ where } \varepsilon \text{ is } 0.$$

In this example, the 3<sup>th</sup> column is linearly dependent on the first two columns when  $\varepsilon = 0$ .

Step 1. Input  $m = 5, n = 3, A$ . Set tolerance as  $10^{-8}$ . The tolerance should be a small number against which to test the length of any  $\alpha_k$  to determine whether  $a_k$  is linearly independent of  $a_1, a_2, \dots, a_{k-1}$  or not.

Step 2. Sweep forward from column 1 through  $n$  and store the  $n$   $\alpha$  and  $n$   $\beta$  vectors

When  $k = 1$ :

(1) Initialize  $\alpha_1, Q_1$ , and  $R_1$

$$\alpha_1 = a_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix},$$

$$Q_1 = I - a_1 a_1^+ = \begin{bmatrix} 0.98181818 & -0.03636364 & -0.05454545 & -0.07272727 & -0.09090909 \\ -0.03636364 & 0.92727273 & -0.10909091 & -0.14545454 & -0.18181818 \\ -0.05454545 & -0.10909091 & 0.83636364 & -0.21818182 & -0.27272727 \\ -0.07272727 & -0.14545454 & -0.21818182 & 0.70909091 & -0.36363636 \\ -0.09090909 & -0.18181818 & -0.27272727 & -0.36363636 & 0.54545454 \end{bmatrix}$$

$$R_1 = (a_1^+)^T a_1^+$$

$$= \begin{bmatrix} 0.00033058 & 0.00066116 & 0.00099174 & 0.00132231 & 0.00165289 \\ 0.00066116 & 0.00132231 & 0.00198347 & 0.00264463 & 0.00330578 \\ 0.00099174 & 0.00198347 & 0.00297521 & 0.00396694 & 0.00495868 \\ 0.00132231 & 0.00264463 & 0.00396694 & 0.00528926 & 0.00661157 \\ 0.00165289 & 0.00330578 & 0.00495868 & 0.00661157 & 0.00826446 \end{bmatrix}$$

When  $k = 2$ :

(1) Compute  $\alpha_2$ :

$$\alpha_2 = Q_1 a_2 = \begin{bmatrix} 3.63636363 \\ 2.27272727 \\ 0.90909091 \\ -0.45454546 \\ -1.81818182 \end{bmatrix},$$

(2) Test length of  $\alpha_2$  against a tolerance

$$|\alpha_2| = 4.76731295,$$

which is greater than the tolerance  $10^{-8}$ . Therefore, Eqs. (2.49), and (2.50) are used to compute

$Q_2$  and  $R_2$ . No  $\beta_2$  is needed.

$$Q_2 = Q_1 - \alpha_2 \alpha_2^+ = \begin{bmatrix} 0.40000000 & -0.40000000 & -0.20000000 & 0.00000000 & 0.20000000 \\ -0.40000000 & 0.70000000 & -0.20000000 & -0.10000000 & 0.00000000 \\ -0.20000000 & -0.20000000 & 0.80000000 & -0.20000000 & -0.20000000 \\ 0.00000000 & -0.10000000 & -0.20000000 & 0.70000000 & -0.40000000 \\ 0.20000000 & 0.00000000 & -0.20000000 & -0.40000000 & 0.40000000 \end{bmatrix}$$

$$R_2 = (\alpha_2^+)^T \alpha_2^+ + (I - a_2 \alpha_2^+)^T R_1 (I - a_2 \alpha_2^+)$$

$$= \begin{bmatrix} 0.15520000 & 0.08800000 & 0.02080000 & -0.04640000 & -0.11360000 \\ 0.08800000 & 0.05000000 & 0.01200000 & -0.02600000 & -0.06400000 \\ 0.02080000 & 0.01200000 & 0.00320000 & -0.00560000 & -0.01440000 \\ -0.04640000 & -0.02600000 & -0.00560000 & 0.01480000 & 0.03520000 \\ -0.11360000 & -0.06400000 & -0.01440000 & 0.03520000 & 0.08480000 \end{bmatrix}.$$

$$A_2^+ = \begin{pmatrix} A_1^+ - A_1^+ a_2 \alpha_2^+ \\ \alpha_2^+ \end{pmatrix},$$

$$= \begin{bmatrix} -0.36000000 & -0.20000000 & -0.04000000 & 0.12000000 & 0.28000000 \\ 0.16000000 & 0.10000000 & 0.04000000 & -0.02000000 & -0.08000000 \end{bmatrix}.$$

When  $k = 3$ :

(1) Compute  $\alpha_3$ :

$$\alpha_3 = Q_2 a_3 = \begin{bmatrix} 0.00000000 \\ 0.00000000 \\ 0.00000000 \\ 0.00000000 \\ 0.00000000 \end{bmatrix},$$

(2) Test length of  $\alpha_3$  against a tolerance

$$|\alpha_3| = 0.00000000,$$

which is less than the tolerance  $10^{-8}$ . Therefore, Eqs. (2.44), (2.45), and (2.46) are used to compute  $Q_3$ ,  $\beta_3$  and  $R_3$ .

$$Q_3 = Q_2 = \begin{bmatrix} 0.40000000 & -0.40000000 & -0.20000000 & 0.00000000 & 0.20000000 \\ -0.40000000 & 0.70000000 & -0.20000000 & -0.10000000 & 0.00000000 \\ -0.20000000 & -0.20000000 & 0.80000000 & -0.20000000 & -0.20000000 \\ 0.00000000 & -0.10000000 & -0.20000000 & 0.70000000 & -0.40000000 \\ 0.20000000 & 0.00000000 & -0.20000000 & -0.40000000 & 0.40000000 \end{bmatrix}$$

$$\beta_3 = R_2 a_3 = \begin{bmatrix} 0.68000000 \\ 0.40000000 \\ 0.12000000 \\ 0.16000000 \\ 0.44000000 \end{bmatrix}$$

$$R_3 = R_2 - \frac{\beta_3 \beta_3^T}{1 + a_k^T \beta_k},$$

$$= \begin{bmatrix} 0.07813333 & 0.04266667 & 0.00720000 & -0.02826667 & -0.06373333 \\ 0.04266667 & 0.02333333 & 0.00400000 & -0.01533333 & -0.03466667 \\ 0.00720000 & 0.00400000 & 0.00080000 & -0.00240000 & -0.00560000 \\ -0.02826667 & -0.01533333 & -0.00240000 & 0.01053333 & 0.02346667 \\ -0.06373333 & -0.03466667 & -0.00560000 & 0.02346667 & 0.05253333 \end{bmatrix}.$$

$$v = \frac{\beta_3}{1 + a_3^T \beta_3} = \begin{bmatrix} 0.11333333 \\ 0.06666667 \\ 0.02000000 \\ -0.02666667 \\ -0.07333333 \end{bmatrix}.$$

$$A_3^+ = \begin{pmatrix} A_2^+ - A_2^+ a_3 v^T \\ v^T \end{pmatrix},$$

$$= \begin{bmatrix} 0.24666667 & -0.13333333 & -0.02000000 & 0.09333333 & 0.20666667 \\ -0.06666667 & -0.03333333 & 0.00000000 & 0.03333333 & 0.06666667 \\ 0.11333333 & 0.06666667 & 0.02000000 & -0.02666667 & -0.07333333 \end{bmatrix}.$$

The generalized inverse calculated by the  $\alpha QR$  algorithm is

$$Ainv = \begin{bmatrix} 0.24666667 & -0.13333333 & -0.02000000 & 0.09333333 & 0.20666667 \\ -0.06666667 & -0.03333333 & 0.00000000 & 0.03333333 & 0.06666667 \\ 0.11333333 & 0.06666667 & 0.02000000 & -0.02666667 & -0.07333333 \end{bmatrix}.$$

To check if the calculations, we compare the generalized inverse with the results given by the function `pinv( )` in MATLAB. The calculation of  $Q$  and  $R$  can also be used to check the correctness of the algorithm. See Eqs. (4.11) and (4.15).

### Example 2

Let us look at an example where the matrix  $A$  is of full rank.

$$A = \begin{bmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 + \varepsilon \end{bmatrix}, \text{ where } \varepsilon \text{ is } 5.$$

In this example, the 3<sup>th</sup> column is linearly independent of the first two columns. Input  $m = 5, n = 3$ . Set tolerance as  $10^{-8}$ . The result of using the  $\alpha Q\beta R$  algorithm to get the generalized inverse is

$$A_{inv} = \begin{bmatrix} -0.40000000 & -0.20000000 & 0.00000000 & 0.20000000 & 0.20000000 \\ -0.00000000 & 0.10000000 & 0.20000000 & 0.30000000 & -0.40000000 \\ 0.10000000 & 0.00000000 & -0.10000000 & -0.20000000 & 0.20000000 \end{bmatrix},$$

which is consistent with the result given by MATLAB.

Example 3

Let us look at an example where the columns in matrix  $A$  are nearly dependent.

$$A = \begin{bmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 + \varepsilon \end{bmatrix}, \text{ where } \varepsilon \text{ is } 10^{-5}.$$

In this example, the 3<sup>rd</sup> column is almost linearly dependent on the first two columns. We want to examine how sensitive this dynamic procedure is to making the decision on whether the matrix is full rank or not. Input  $m = 5, n = 3$ . First, the third column is treated as dependent on the first two columns. The result of using the  $\alpha Q\beta R$  algorithm to get the generalized inverse is

$$A_{inv} = \begin{bmatrix} -0.24666701 & -0.13333353 & -0.02000005 & 0.09333343 & 0.20666690 \\ -0.06666653 & -0.03333326 & 0.00000000 & 0.03333327 & 0.06666654 \\ 0.11333331 & 0.06666666 & 0.02000000 & -0.02666665 & -0.07333330 \end{bmatrix},$$

which is not consistent with the result given by MATLAB. Now let the third column be treated as independent of the first two columns. The result of using the  $\alpha Q\beta R$  algorithm to get the generalized inverse is

$$A_{inv} = 1.0e5 \begin{bmatrix} -0.49999500 & -0.00000200 & -0.49999900 & 0.99999600 & 1.00000000 \\ -0.99999800 & -0.00000100 & 1.00000000 & 1.99999900 & -2.00000000 \\ 0.50000000 & 0.00000000 & -0.50000000 & -1.00000000 & 1.00000000 \end{bmatrix},$$

which is consistent to the digits shown with the result given by MATLAB. This seems to indicate that making the correct decision is very important.

## 6. CONCLUSIONS

The purpose of this paper is to show the connection between the  $\alpha Q\beta R$  algorithm and the algorithm of Greville. When carrying out the  $\alpha Q\beta R$  algorithm, one can produce enough information to calculate the generalized inverse. Many numerical experiments – of which only a few are reported here – show that the  $\alpha Q\beta R$  algorithm is reliable and efficient.

From the previous discussion, the meaning of the  $\alpha$ 's and the  $Q$ 's is clear. But the meaning of the  $\beta$ 's and the  $R$ 's is more obscure. Let us write Eq. (4.13) as

$$\beta_k = v(1 + a_k^T \beta_k).$$

We do know that  $\beta_k$  is proportional to  $v$ , which appears in Greville's updating formula, because  $(1 + a_k^T \beta_k)$  is just a scalar.

## References

1. T. Kohonen, 1989. *Self-Organization & Associative Memory*. Springer-Verlag New York, Inc.

2. Bohrnstedt, Geoge W and David Knoke, 1994. *Statistics for Social Data Analysis*. Third Edition. Itasca, III.: F. E. Peacock.
3. F. Udwwadia and R. Kalaba, *Analytical Dynamics*, 1996, Cambridge Univ. Press, London.
4. R. Kalaba and N. Rasakhoo, *Algorithms for generalized inverses*, Journal of Optimization Theory and Applications, vol. 48, no. 3, pp. 427-435, 1986.
5. F. Udwwadia and R. Kalaba, *An Alternative Proof of the Greville Formula*, Journal of Optimization Theory and Applications, vol. 94, no. 3, pp. 23-28, 1997.
6. R. Bellman and R. Kalaba, *Dynamic Programming and Modern Control Theory*, McGraw-Hill, N.Y., 1965.
7. R. Kalaba, R. Xu. and W. Feng, *Solving Shortest Length Least-Squares Problems via Dynamic Programming*, Journal of Optimization Theory and Applications, Vol. 85, No. 3, pp. 613-632, 1995.
8. R. Kalaba and H. Natsuyama, *Dynamic Programming and Minimal Norm solutions of Least Squares Problems*, submitted for publication.

**APPENDIX I**

Below is the program for this algorithm for both solving least square problems and calculating generalized inverses. It is compiled and run in MatLab version 5.3.

```

diary result1.txt
clear all
%echo on
%aQBR Algorithm
%Yueyue Fan 10-1-2000

%input the A matrix and b vector
A = input('Please input A:')
M = input('Please input the dimension M:')
N = input('Please input the dimension N:')
e = input('Please input the tolerance:')

%initialize alfa1, Q1, and R1
a = A(:,1);
Ak = a;
pinvAk = Ak'/(Ak'*Ak);
alfa(:,1) = a
I = eye(M);
pinva = a'/(a'*a)
Q = I - a*pinva
R = pinva'*pinva
Beta(:,1) = zeros(M,1)
AAk = a
Ainv = AAk'/(AAk'*AAk)
for k = 2 : N
    disp(k);
    alfa(:,k) = Q * A(:,k)
    AAk = A(:,[1:k])
    %Test length of alfa(k) against a tolerance and calculate alfa, Q, R, Beta
    for each k.
        length = norm(alfa(:,k))
        if length <= e
            Q = Q
            Beta(:,k) = R * A(:,k)
            R = R - Beta(:,k)*Beta(:,k)'/(1+A(:,k)'*Beta(:,k))
            v = Beta(:,k) / (1 + A(:,k)' * Beta(:,k));
            AinvTop = Ainv - Ainv * A(:,k) * v';
            Ainv = AinvTop;
            Ainv(k,:) = v'
        else
            pinvAlfaK = alfa(:,k)'/(alfa(:,k)'*alfa(:,k));
            Q = Q - alfa(:,k) * pinvAlfaK

            RR = I - A(:,k) * pinvAlfaK
            R = pinvAlfaK' * pinvAlfaK + RR' * R * RR
            AinvTop = Ainv - Ainv * A(:,k) * pinvAlfaK;
            Ainv = AinvTop;
            Ainv(k,:) = pinvAlfaK
        end
    end
end

```

```
end

%output
disp('The Generalized Inverse of Matrix A is:');
disp(Ainv);
disp('END');
%echo off
diary off;
```

**APPENDIX II**

The procedure for proving the two relationships, (1)  $\alpha_k$  is orthogonal to  $\alpha_1, \alpha_2, \dots, \alpha_{k-1}$ ;

(2)  $a_k = \alpha_k + \text{lin. com. of } \alpha_1, \alpha_2, \dots, \alpha_{k-1}$ , where  $\alpha_k = Q_{k-1}a_k$ , is given below.

The initial conditions are:

$$\alpha_1 = a_1,$$

$$Q_1 = I - \frac{\alpha_1 \alpha_1^T}{\alpha_1^T \alpha_1} = I - a_1 a_1^+.$$

It follows that

$$Q_1 a_1 = I a_1 - \frac{\alpha_1 \alpha_1^T a_1}{\alpha_1^T \alpha_1} = 0,$$

$$Q_1 v = I v - \frac{\alpha_1 \alpha_1^T v}{\alpha_1^T \alpha_1} = v,$$

where  $v$  is orthogonal to  $\alpha_1$ .

When  $k = 2$ :

$$\begin{aligned} \alpha_2 \alpha_1 &= (Q_1 a_2)^T a_1, \\ &= a_2^T Q_1 a_1, \\ &= 0. \end{aligned}$$

Therefore,  $\alpha_2$  is orthogonal to  $a_1$ . The first relationship holds when  $k = 2$ .

$$\begin{aligned} a_2 - \alpha_2 &= a_2 - Q_1 a_2, \\ &= (I - I + \frac{\alpha_1 \alpha_1^T}{\alpha_1^T \alpha_1}) a_2, \\ &= (\frac{\alpha_1^T a_2}{\alpha_1^T \alpha_1}) \alpha_1. \end{aligned}$$

Because  $\left(\frac{\alpha_1^T a_2}{a_1^T \alpha_1}\right)$  is a certain scalar, and  $\alpha_1 = a_1$ , we have

$$a_2 - \alpha_2 = s \cdot a_1,$$

where  $s$  is a scalar. Therefore, when  $k = 2$ , the second relationship also holds.

Next, let us assume the two relationships hold for the 1<sup>st</sup>, 2<sup>nd</sup>, ...,  $k^{\text{th}}$  cases; namely, (1)  $\alpha_l$  is orthogonal to  $\alpha_1, \alpha_2, \dots, \alpha_{l-1}$ ; (2)  $\alpha_l = \alpha_l + \text{lin. com. of } \alpha_1, \alpha_2, \dots, \alpha_{l-1}$ .

By definition, we have:

$$Q_l = I - \frac{\alpha_1 \alpha_1^T}{a_1^T \alpha_1} - \dots - \frac{\alpha_{l-1} \alpha_{l-1}^T}{a_{l-1}^T \alpha_{l-1}}, \text{ and}$$

$$\alpha_l = Q_{l-1} a_l,$$

where  $l = 1, 2, \dots, k$ .

Because

$$a_l^T \alpha_l = (\alpha_l + \text{lin. com. of } \alpha_1, \alpha_2, \dots, \alpha_{l-1})^T \alpha_l, \text{ and}$$

$$\alpha_i^T \alpha_l = 0, \text{ where } i < l,$$

it follows that

$$a_l^T \alpha_l = \alpha_l^T \alpha_l.$$

Therefore,  $Q_l$  can be also written as:

$$Q_l = I - \frac{\alpha_1 \alpha_1^T}{\alpha_1^T \alpha_1} - \dots - \frac{\alpha_{l-1} \alpha_{l-1}^T}{\alpha_{l-1}^T \alpha_{l-1}}.$$

Now let us look at the  $k+1^{\text{th}}$  case:

$$\begin{aligned} \alpha_{k+1}^T \alpha_l &= (Q_k a_{k+1})^T \alpha_l \\ &= a_{k+1}^T Q_k \alpha_l \end{aligned}$$

$$= a_{k+1}^T \left( I - \frac{\alpha_1 \alpha_1^T}{\alpha_1^T \alpha_1} - \dots - \frac{\alpha_l \alpha_l^T}{\alpha_l^T \alpha_l} - \dots - \frac{\alpha_k \alpha_k^T}{\alpha_k^T \alpha_k} \right) \alpha_l.$$

Because  $\alpha_i^T \alpha_l = 0$ , where  $i \neq l$ , it is obvious that

$$\alpha_{k+1}^T \alpha_l = a_{k+1}^T \left( \alpha_l - \frac{\alpha_l \alpha_l^T}{\alpha_l^T \alpha_l} \alpha_l \right) = 0.$$

It has been proved that  $\alpha_{k+1}$  is orthogonal to the previous  $\alpha$ 's. Therefore, the first relationship holds for the  $k+1^{th}$  case. Also,

$$\begin{aligned} a_{k+1} - \alpha_{k+1} &= a_{k+1} - Q_k a_{k+1} \\ &= (I - Q_k) a_{k+1} \\ &= \left( \frac{\alpha_1 \alpha_1^T}{\alpha_1^T \alpha_1} + \dots + \frac{\alpha_k \alpha_k^T}{\alpha_k^T \alpha_k} \right) a_{k+1} \\ &= \text{lin. com. of } \alpha_1 \cdots \alpha_k. \end{aligned}$$

Therefore, the second relationship also holds for the  $(k+1)^{st}$  case. This completes the proofs.