Available online at www.sciencedirect.com
ScienceDirect
Applied Numerical Mathematics 57 (2007) 1240–1244
APPLIED NUMERICAL MATHEMATICS
www.elsevier.com/locate/apnum

Euclidean distances and least squares problems for a given set of vectors ☆

Xiao-Wen Chang *, Christopher C. Paige

School of Computer Science, McGill University, Montreal, Quebec, Canada, H3A 2A7

Available online 9 February 2007

Abstract

Given an $m \times n$ matrix $\boldsymbol{A}$, n Euclidean distances, those from each column to the space spanned by the remaining columns of $\boldsymbol{A}$, are considered. An elegant relationship between $\boldsymbol{A}$, these Euclidean distances, and the solutions of n simple linear least squares problems arising from $\boldsymbol{A}$ is derived. When $\boldsymbol{A}$ has full column rank, from this a useful expression for these Euclidean distances is immediately obtained. The theory is then used to greatly improve the efficiency of an algorithm used in communications.

Keywords: Least squares; Residuals; Euclidean distance; QR factorization; Permutations; Communications

1. Introduction

Given n real m-vectors, n simple linear least squares (LS) problems can be formed, each using one of these n vectors as the right-hand side of a LS problem, while the remaining $n-1$ vectors form the coefficient matrix of the problem. The 2-norm of each least squares residual vector is the Euclidean distance from the chosen right-hand side vector to the space spanned by the other $n-1$ vectors.

This paper shows an elegant relationship between these Euclidean distances (2-norms of the LS residuals) and the LS solutions, and gives a general formula for each Euclidean distance. Then an application in communications is considered, in which the QR factorization of a certain column permutation of an $m \times n$ matrix is needed. Based on the formulae for the Euclidean distances, a greatly improved algorithm is proposed for finding the permutations and obtaining the QR factorization.

In the paper bold upper case letters and bold lower case letters are used to denote matrices and vectors respectively. The identity matrix is denoted by $\boldsymbol{I}$ and its ith column by $\boldsymbol{e}_i$. Sometimes MATLAB notation is used to denote a submatrix. Specifically, if $\boldsymbol{A}$ is a matrix, then $\boldsymbol{A}(:, j)$ is the jth column of $\boldsymbol{A}$, $\boldsymbol{A}(:, j_1 : j_2)$ denotes the submatrix formed by columns j_1 to j_2, and $\boldsymbol{A}(i_1 : i_2, j_1 : j_2)$ the submatrix formed by rows i_1 to i_2 and columns j_1 to j_2.

☆ This research was supported by NSERC of Canada Grants RGPIN217191-03 for Xiao-Wen Chang and RGPIN9236-01 for Christopher C. Paige.
* Corresponding author.
E-mail addresses: chang@cs.mcgill.ca (X.-W. Chang), paige@cs.mcgill.ca (C.C. Paige).
URLs: http://www.cs.mcgill.ca/~chang (X.-W. Chang), http://www.cs.mcgill.ca/~chris (C.C. Paige).

0168-9274/$30.00
doi:10.1016/j.apnum.2007.01.008

2. The main result

The following theorem gives some interesting and useful general results.

Theorem 2.1. *Given an $m \times n$ real matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n]$, let $\boldsymbol{D}(\boldsymbol{A}) \equiv \operatorname{diag}(\delta_1(\boldsymbol{A}), \ldots, \delta_n(\boldsymbol{A}))$ where for $i = 1, \ldots, n$, $\delta_i(\boldsymbol{A})$ is the Euclidean distance from $\boldsymbol{a}_i$ to the space spanned by all other columns of $\boldsymbol{A}$, i.e.,*

$$\delta_i(\boldsymbol{A}) = \min_{\boldsymbol{x}_i} \left\| \boldsymbol{a}_i - [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{i-1}, \boldsymbol{a}_{i+1}, \ldots, \boldsymbol{a}_n]\boldsymbol{x}_i \right\|_2 = \|\boldsymbol{A}\bar{\boldsymbol{x}}_i\|_2.$$

Here if $\hat{\boldsymbol{x}}_i \equiv [\hat{x}_{1,i}, \ldots, \hat{x}_{i-1,i}, \hat{x}_{i+1,i}, \ldots, \hat{x}_{n,i}]^T$ is any vector solving the above LS problem ($\hat{\boldsymbol{x}}_i$ might not be unique), we define

$$\bar{\boldsymbol{x}}_i \equiv [\hat{x}_{1,i}, \ldots, \hat{x}_{i-1,i}, -1, \ \hat{x}_{i+1,i}, \ldots, \hat{x}_{n,i}]^T, \qquad \bar{\boldsymbol{X}} \equiv [\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_2, \ldots, \bar{\boldsymbol{x}}_n]. \tag{1}$$

Then there is a relationship between $\boldsymbol{A}$, the "solution" vectors $\bar{\boldsymbol{x}}_i$, and the Euclidean distances $\delta_i(\boldsymbol{A})$:

$$\boldsymbol{A}^T\boldsymbol{A}\bar{\boldsymbol{X}} = -\boldsymbol{D}^2. \tag{2}$$

When $\boldsymbol{A}$ has full column rank, the diagonal elements of the symmetric matrix $\bar{\boldsymbol{X}}\boldsymbol{D}^{-2} = -(\boldsymbol{A}^T\boldsymbol{A})^{-1}$ show that

$$\delta_i(\boldsymbol{A})^2 = \frac{1}{\boldsymbol{e}_i^T(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{e}_i}, \quad i = 1, 2, \ldots, n, \tag{3}$$

while for $i, j = 1, 2, \ldots, n$, with $i \neq j$, the off-diagonal elements give

$$\frac{\hat{x}_{j,i}}{\delta_i(\boldsymbol{A})^2} = \frac{\hat{x}_{i,j}}{\delta_j(\boldsymbol{A})^2} = -\boldsymbol{e}_i^T\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{e}_j. \tag{4}$$

This relates the least squares residual norms (or Euclidean distances) $\delta_i(\boldsymbol{A})$ and $\delta_j(\boldsymbol{A})$, to the least squares solutions $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$. Combining (3) *and* (4) *relates the least squares solutions $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$ to the elements of $(\boldsymbol{A}^T\boldsymbol{A})^{-1}$:*

$$\hat{x}_{j,i}\boldsymbol{e}_i^T\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{e}_i = \hat{x}_{i,j}\boldsymbol{e}_j^T\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{e}_j = -\boldsymbol{e}_i^T\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{e}_j. \tag{5}$$

Proof. With the definitions in (1), for $i = 1, \ldots, n$ denote

$$\boldsymbol{A}_i \equiv [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{i-1}, \boldsymbol{a}_{i+1}, \ldots, \boldsymbol{a}_n], \qquad \boldsymbol{d}_i \equiv \boldsymbol{a}_i - \boldsymbol{A}_i\hat{\boldsymbol{x}}_i = -\boldsymbol{A}\bar{\boldsymbol{x}}_i. \tag{6}$$

The normal equations for the LS problem $\min_{\boldsymbol{x}_i}\|\boldsymbol{a}_i - \boldsymbol{A}_i\boldsymbol{x}_i\|_2$ give

$$\boldsymbol{A}_i^T\boldsymbol{d}_i = \boldsymbol{0}. \tag{7}$$

Thus from (6) and (7) it follows that

$$\boldsymbol{a}_i^T\boldsymbol{d}_i = (\boldsymbol{d}_i + \boldsymbol{A}_i\hat{\boldsymbol{x}}_i)^T\boldsymbol{d}_i = \boldsymbol{d}_i^T\boldsymbol{d}_i = \delta_i(\boldsymbol{A})^2. \tag{8}$$

Combining (7) and (8), we obtain

$$\boldsymbol{A}^T\boldsymbol{d}_i = \boldsymbol{e}_i\delta_i(\boldsymbol{A})^2,$$

which, by the expression for $\boldsymbol{d}_i$ in (6), is equivalent to

$$\boldsymbol{A}^T\boldsymbol{A}\bar{\boldsymbol{x}}_i = -\boldsymbol{e}_i\delta_i(\boldsymbol{A})^2.$$

But this gives (2); and then (3), (4) and (5) follow for full column rank $\boldsymbol{A}$. □

Corollary 2.1. *Let full column rank $\boldsymbol{A}$ have the QR factorization*

$$\boldsymbol{A} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}, \tag{9}$$

where $\boldsymbol{Q}$ is an $m \times m$ orthogonal matrix and $\boldsymbol{R}$ is an $n \times n$ nonsingular upper triangular matrix with elements r_{ij}. Then

$$r_{kk}^2 = \delta_k\big(\boldsymbol{A}(:, 1:k)\big)^2 = \frac{1}{\boldsymbol{e}_k^T(\boldsymbol{A}(:, 1:k)^T\boldsymbol{A}(:, 1:k))^{-1}\boldsymbol{e}_k}, \quad k = 1, \ldots, n. \tag{10}$$

Proof. From the upper triangular structure of $\boldsymbol{R}$ it can be seen that $|r_{kk}|$ is the Euclidean distance from $\boldsymbol{R}(:,k)$ to the space spanned by the columns of $\boldsymbol{R}(:,1:k-1)$, or equivalently the Euclidean distance from $\boldsymbol{a}_k$ to the space spanned by $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_{k-1}$. Thus the first equality in (10) holds. The second equality in (10) follows from (3) in Theorem 2.1. □

Remark 2.1. The formula (3) in Theorem 2.1 can be proved in a slightly more straightforward way. Suppose we interchange the ith column and the nth column of full column rank $\boldsymbol{A}$ and find the corresponding QR factorization:

$$\boldsymbol{A}\boldsymbol{P} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}, \qquad \boldsymbol{P} \equiv [\boldsymbol{e}_1,\ldots,\boldsymbol{e}_{i-1},\boldsymbol{e}_n,\boldsymbol{e}_{i+1},\ldots,\boldsymbol{e}_{n-1},\boldsymbol{e}_i].$$

Then

$$\boldsymbol{P}^T\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{P} = \boldsymbol{R}^{-1}\boldsymbol{R}^{-T}.$$

Comparing the (n,n) elements of both sides, we obtain

$$\boldsymbol{e}_n^T\boldsymbol{P}^T\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{P}\boldsymbol{e}_n = 1/r_{nn}^2.$$

But from the upper triangular structure of $\boldsymbol{R}$, we see that $|r_{nn}|$ is the Euclidean distance from the last column of $\boldsymbol{A}\boldsymbol{P}$ (which is $\boldsymbol{a}_i$) to the space spanned by the other columns of $\boldsymbol{A}\boldsymbol{P}$, so $|r_{nn}| = \delta_i(\boldsymbol{A})$. Thus

$$\delta_i(\boldsymbol{A})^2 = 1/\boldsymbol{e}_i^T\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{e}_i.$$

Remark 2.2. The relation between $|r_{kk}|$ and $\delta_k(\boldsymbol{A}(:,1:k))$ given by the first equality in (10) is known, see [2, p. 104].

3. An application

The theory given in Section 2 might have several uses. Here we show one use for (3).

In solving integer linear least squares problems which may arise in decoding or detection in communications, one often computes the QR factorization of $\boldsymbol{A}$ with column pivoting in order to make the later computation (a search process) efficient (see, e.g., [1] and [3]). The permutation matrix $\boldsymbol{P}$ is usually determined such that the $|r_{kk}|$ are large for large k and small for small k (note that $|r_{11}r_{22}\cdots r_{nn}|$ is fixed). In [3], the so-called vertical Bell Labs layered space-time (V-BLAST) optical detection ordering given in [4] was proposed for the column permutations. This permutation strategy determines the columns of the permuted matrix $\boldsymbol{A}\boldsymbol{P}$ from the last to the first. Let $\mathcal{J}_k$ denote the set of column indices for the not yet chosen columns when the kth column of the permuted matrix $\boldsymbol{A}\boldsymbol{P}$ is to be determined (where $k = n, n-1,\ldots,1$). Then this strategy chooses the $p(k)$th column of the original matrix $\boldsymbol{A}$ to be the kth column of the permuted matrix $\boldsymbol{A}\boldsymbol{P}$ that we seek, as follows:

$$p(k) = \arg\max_{j\in\mathcal{J}_k} \boldsymbol{a}_j^T\left[\boldsymbol{I} - \boldsymbol{A}_{k,j}\left(\boldsymbol{A}_{k,j}^T\boldsymbol{A}_{k,j}\right)^{-1}\boldsymbol{A}_{k,j}^T\right]\boldsymbol{a}_j, \tag{11}$$

where $\boldsymbol{A}_{k,j}$ is the $m\times(k-1)$ matrix formed by the columns $\boldsymbol{a}_i$ with $i\in\mathcal{J}_k - \{j\}$. We can easily show that $\boldsymbol{a}_j^T[\boldsymbol{I} - \boldsymbol{A}_{k,j}(\boldsymbol{A}_{k,j}^T\boldsymbol{A}_{k,j})^{-1}\boldsymbol{A}_{k,j}^T]\boldsymbol{a}_j$ in (11) is the squared Euclidean distance from $\boldsymbol{a}_j$ to the space spanned by the columns of $\boldsymbol{A}_{k,j}$. But Corollary 2.1 tells us that $|r_{kk}|$ is the Euclidean distance from the kth column of $\boldsymbol{A}\boldsymbol{P}$ to the space spanned by the first $k-1$ columns of $\boldsymbol{A}\boldsymbol{P}$. Thus for $k = n, n-1,\ldots,1$, $\boldsymbol{a}_{p(k)}$ is the column which makes $|r_{kk}|$ maximum among all the not yet chosen columns when we determine the kth column of the permuted matrix $\boldsymbol{A}\boldsymbol{P}$, i.e., the resulting $|r_{kk}|$ is the largest which can be found at this step.

The following is an obvious way to find this permutation matrix $\boldsymbol{P}$. We first compute the QR factorization of $\boldsymbol{A}$ as in (9) and record $|r_{nn}|$. In order to determine the last column of the permuted $\boldsymbol{A}$, we interchange the last column of $\boldsymbol{R}$ with its jth column for $j = 1,2,\ldots,n-1$. After each interchange, we compute the QR factorization of the permuted $\boldsymbol{R}$, which can be done in an efficient way (taking $O(n^2)$ flops) by using the structure of the permuted $\boldsymbol{R}$, and record the corresponding $|r_{nn}|$. By comparing the n possible values of $|r_{nn}|$, we determine the desired last column of the permuted $\boldsymbol{A}$, and keep the updated $\boldsymbol{R}$-factor for use in the next step. We see that the cost of these computations is $O(n^3)$ flops. Then in order to determine the desired second to last column of the permuted $\boldsymbol{A}$, we apply a similar process to the $(n-1)\times(n-1)$ submatrix $\boldsymbol{R}(1:n-1,1:n-1)$ of the updated $\boldsymbol{R}$ we have just obtained — to obtain a new updated upper triangular $\boldsymbol{R}(1:n-1,1:n-1)$ having the largest $|r_{n-1,n-1}|$. After repeating the above process $n-1$ times, all the permutations are determined.

For the final $\boldsymbol{R}$ we hope that $0 < |r_{11}| \leqslant \cdots \leqslant |r_{nn}|$, but this is not necessarily true as the example $\boldsymbol{A} = \begin{bmatrix} 9 & 6 \\ 0 & 8 \end{bmatrix}$ shows, since exchanging the columns leads to $|r_{22}| = 7.2$, showing that $\boldsymbol{A}$ is already in the chosen final order.

The above algorithm is numerically stable, but it needs $O(n^4)$ flops. Later a faster algorithm is given. For motivation consider the QR factorization of any permutation of the columns of $\boldsymbol{A}$, and its first k columns,

$$\boldsymbol{AP} = \boldsymbol{Q}\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix}, \qquad \boldsymbol{AP}(:, 1:k) = \boldsymbol{Q}(:, 1:k)\boldsymbol{R}(1:k, 1:k). \tag{12}$$

From this and Corollary 2.1 we see that if $\boldsymbol{P}$ is the permutation matrix arising from V-BLAST, then for $k = n, n-1, \ldots, 1$,

$$|r_{kk}| = \delta_k\big(\boldsymbol{AP}(:, 1:k)\big) = \max_{1 \leqslant j \leqslant k} \delta_j\big(\boldsymbol{AP}(:, 1:k)\big),$$

or equivalently, using (3), the aim of V-BLAST is to find $\boldsymbol{P}$ such that

$$\boldsymbol{e}_k^T\big[\boldsymbol{P}(:, 1:k)^T\boldsymbol{A}^T\boldsymbol{AP}(:, 1:k)\big]^{-1}\boldsymbol{e}_k = \min_{1 \leqslant j \leqslant k} \boldsymbol{e}_j^T\big[\boldsymbol{P}(:, 1:k)^T\boldsymbol{A}^T\boldsymbol{AP}(:, 1:k)\big]^{-1}\boldsymbol{e}_j. \tag{13}$$

For the faster algorithm, first compute the QR factorization of $\boldsymbol{A}$ and denote this

$$\boldsymbol{A} = \bar{\boldsymbol{Q}}\begin{bmatrix} \bar{\boldsymbol{R}} \\ \boldsymbol{0} \end{bmatrix}. \tag{14}$$

Next compute $\bar{\boldsymbol{L}} \equiv \bar{\boldsymbol{R}}^{-T}$, which takes about $n^3/3$ flops (cf. [2, p. 119]). Note for the diagonal elements, $\bar{l}_{ii}\bar{r}_{ii} = 1$, and from (3)

$$\delta_n(\boldsymbol{A})^{-2} = \boldsymbol{e}_n^T\big(\boldsymbol{A}^T\boldsymbol{A}\big)^{-1}\boldsymbol{e}_n = \boldsymbol{e}_n^T\bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{R}}^{-T}\boldsymbol{e}_n = \boldsymbol{e}_n^T\bar{\boldsymbol{L}}^T\bar{\boldsymbol{L}}\boldsymbol{e}_n = \bar{l}_{nn}^2 = \bar{r}_{nn}^{-2},$$

so if $\boldsymbol{L} = \boldsymbol{R}^{-T}$ in (12), to maximize $\delta_n(\boldsymbol{AP})$ over all permutations $\boldsymbol{P}$ we can either maximize r_{nn}^2, or minimize l_{nn}^2 (and so on for further steps). To do this, we compute a QL factorization of $\bar{\boldsymbol{L}}$ with a particular form of pivoting:

$$\bar{\boldsymbol{L}}\bar{\boldsymbol{P}} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{L}}. \tag{15}$$

This is done using permutations and orthogonal rotations, and will cost no more than about n^3 flops (note that this does not count the cost of explicitly forming $\tilde{\boldsymbol{Q}}$, since it is not necessary in the application). We just work on $\bar{\boldsymbol{L}}$ until finally it becomes $\tilde{\boldsymbol{L}}$. In the first step we move the column of $\bar{\boldsymbol{L}}$ with smallest 2-norm (suppose it is the pth column) to become the last column of $\bar{\boldsymbol{L}}\bar{\boldsymbol{P}}$ (moving each of columns $p+1, \ldots, n$ down one). We then apply $n-p$ rotations from the left of $\bar{\boldsymbol{L}}\bar{\boldsymbol{P}}$ to bring it back to lower triangular form (cf. [2, p. 134]), which we call 'the updated $\bar{\boldsymbol{L}}$'. In general, for $k = n, n-1, \ldots, 2$, in the $(n-k+1)$th step of the reduction process we determine the index p such that column p of the updated submatrix $\bar{\boldsymbol{L}}(1:k, 1:k)$ has the *smallest* 2-norm among all of its columns. Then we move column p to be the kth column of the updated $\bar{\boldsymbol{L}}$. After that we apply $k-p$ orthogonal rotations to zero $\bar{\boldsymbol{L}}(1:k-1, k)$. This step costs about $3(k-2p+2)k$ flops, from which it can be concluded that the total cost of the algorithm to obtain the QL factorization (15) is at most about n^3 flops (depending on the permutations required). Thus we have for the column norms of $\tilde{\boldsymbol{L}}(1:k, 1:k)$

$$\boldsymbol{e}_k^T\tilde{\boldsymbol{L}}(1:k, 1:k)^T\tilde{\boldsymbol{L}}(1:k, 1:k)\boldsymbol{e}_k = \min_{1 \leqslant j \leqslant k} \boldsymbol{e}_j^T\tilde{\boldsymbol{L}}(1:k, 1:k)^T\tilde{\boldsymbol{L}}(1:k, 1:k)\boldsymbol{e}_j. \tag{16}$$

But since $\bar{\boldsymbol{L}} = \bar{\boldsymbol{R}}^{-T}$, from (15) and (14) it follows that

$$\tilde{\boldsymbol{L}}^{-1}\tilde{\boldsymbol{L}}^{-T} = \bar{\boldsymbol{P}}^T\bar{\boldsymbol{L}}^{-1}\bar{\boldsymbol{L}}^{-T}\bar{\boldsymbol{P}} = \bar{\boldsymbol{P}}^T\bar{\boldsymbol{R}}^T\bar{\boldsymbol{R}}\bar{\boldsymbol{P}} = \bar{\boldsymbol{P}}^T\boldsymbol{A}^T\boldsymbol{A}\bar{\boldsymbol{P}}.$$

Comparing the $k \times k$ leading principal submatrices of both sides, we have

$$\tilde{\boldsymbol{L}}(1:k, 1:k)^{-1}\tilde{\boldsymbol{L}}(1:k, 1:k)^{-T} = \bar{\boldsymbol{P}}(:, 1:k)^T\boldsymbol{A}^T\boldsymbol{A}\bar{\boldsymbol{P}}(:, 1:k).$$

Therefore

$$\tilde{\boldsymbol{L}}(1:k, 1:k)^T\tilde{\boldsymbol{L}}(1:k, 1:k) = \big[\bar{\boldsymbol{P}}(:, 1:k)^T\boldsymbol{A}^T\boldsymbol{A}\bar{\boldsymbol{P}}(:, 1:k)\big]^{-1}.$$

From this and (16) it follows that for $k = n, \ldots, 1$

$$\boldsymbol{e}_k^T\big[\bar{\boldsymbol{P}}(:, 1:k)^T\boldsymbol{A}^T\boldsymbol{A}\bar{\boldsymbol{P}}(:, 1:k)\big]^{-1}\boldsymbol{e}_k = \min_{1 \leqslant j \leqslant k} \boldsymbol{e}_j^T\big[\bar{\boldsymbol{P}}(:, 1:k)^T\boldsymbol{A}^T\boldsymbol{A}\bar{\boldsymbol{P}}(:, 1:k)\big]^{-1}\boldsymbol{e}_j, \tag{17}$$

which with (13) shows that $\bar{\boldsymbol{P}}$ in (15) has the desired properties of $\boldsymbol{P}$ in V-BLAST, and so we have found the permutations far more efficiently.

If $\boldsymbol{A}$ is close to rank deficient, then the numerical computation $\bar{\boldsymbol{L}} \equiv \bar{\boldsymbol{R}}^{-T}$ might occasionally result in $\bar{\boldsymbol{P}}$ in (15) being not quite correct. That is, it might not give (17) for every k. Such an occurrence (hopefully rare in practice) could alter the number of steps in the later search process in solving the integer least squares problem, but would have no other deleterious effects.

So far we have found $\bar{\boldsymbol{P}}$, whereas our ultimate aim was to compute the QR factorization of $\boldsymbol{A}\bar{\boldsymbol{P}}$. There are two obvious ways to do this. From (14), $\bar{\boldsymbol{R}} = \bar{\boldsymbol{L}}^{-T}$, and (15), we obtain

$$\boldsymbol{A}\bar{\boldsymbol{P}} = \bar{\boldsymbol{Q}}\begin{bmatrix}\bar{\boldsymbol{L}}^{-T}\bar{\boldsymbol{P}} \\ \boldsymbol{0}\end{bmatrix} = \bar{\boldsymbol{Q}}\begin{bmatrix}\tilde{\boldsymbol{Q}} & \\ & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{L}}^{-T} \\ \boldsymbol{0}\end{bmatrix},$$

which, with $\boldsymbol{R} \equiv \tilde{\boldsymbol{L}}^{-T}$, gives the QR factorization we have sought:

$$\boldsymbol{A}\bar{\boldsymbol{P}} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R} \\ \boldsymbol{0}\end{bmatrix}, \qquad \boldsymbol{Q} \equiv \bar{\boldsymbol{Q}}\begin{bmatrix}\tilde{\boldsymbol{Q}} & \\ & \boldsymbol{I}\end{bmatrix}. \tag{18}$$

Computing $\boldsymbol{R} \equiv \tilde{\boldsymbol{L}}^{-T}$ again takes about $n^3/3$ flops. The problem is that $\boldsymbol{R}$ may lose accuracy unnecessarily since it is obtained via the computation of the inverses of two triangular matrices. The other approach is to use the permutation matrix $\bar{\boldsymbol{P}}$ obtained in (15) and compute the QR factorization $\bar{\boldsymbol{R}}\bar{\boldsymbol{P}} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$, so that from (14)

$$\boldsymbol{A}\bar{\boldsymbol{P}} = \bar{\boldsymbol{Q}}\begin{bmatrix}\bar{\boldsymbol{R}}\bar{\boldsymbol{P}} \\ \boldsymbol{0}\end{bmatrix} = \bar{\boldsymbol{Q}}\begin{bmatrix}\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}} \\ \boldsymbol{0}\end{bmatrix} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R} \\ \boldsymbol{0}\end{bmatrix}, \qquad \boldsymbol{Q} \equiv \bar{\boldsymbol{Q}}\begin{bmatrix}\hat{\boldsymbol{Q}} & \\ & \boldsymbol{I}\end{bmatrix}, \qquad \boldsymbol{R} \equiv \hat{\boldsymbol{R}},$$

which is a continuation of the original QR factorization (14), and so is also numerically stable. This approach takes at most about n^3 flops (depending on the permutations), which at worst could be a little more costly than the first approach, but it is more numerically reliable and so is more preferable in general.

Acknowledgements

The authors would like to thank Gene Golub and Michael Saunders for their helpful comments. The first author is indebted to them for their hospitality when he was on sabbatical at Stanford University where part of this research work was done.

References

[1] E. Agrell, T. Eriksson, A. Vardy, K. Zeger, Closest point search in lattices, IEEE Trans. Inform. Theory 48 (2002) 2201–2214.
[2] A. Björck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, PA, 1996.
[3] M.O. Damen, H. El Gamal, G. Caire, On maximum-likelihood detection and the search for the closest lattice point, IEEE Trans. Inform. Theory 49 (2003) 2389–2402.
[4] G.J. Foscini, G.D. Golden, R.A. Valenzuela, P.W. Wolniansky, Simplified processing for high spectral efficiency wireless communication employing multi-element arrays, IEEE J. Select. Areas Commun. 17 (1999) 1841–1852.