Compact Schemes for A^+B, A^+AB and AA^+B

Marc Stromberg*

Pacific States Marine Fisheries Commission, U S A.

Corresponding Author: Marc Stromberg, Pacific States Marine Fisheries Commission, U S A.

Abstract

Explicit details are presented for calculation of A^+B, AA^+AB and AA^+B where $A_{m \times n}$ is any nonzero matrix, A^+ is the Moore-Penrose pseudoinverse of A and B is any matrix of appropriate dimensions, where the quantities in question are found using only the storage originally allocated to the matrices A and B (together with some simple one dimensional indexing arrays).

Keywords: Nonzero Matrix, A+B, A+AB, AA+B

1. Introduction

The calculations to be presented depend on the following theorem, which guarantees a specific rank decomposition of any nonzero $m \times n$ matrix. The proof of the theorem is given with minor modifications in Appendix A, as is a statement of the algorithm that produces the factorization of the theorem. The algorithm is typical of so-called compact schemes, in that the entries of the factors are stored by modifying those of A in place. This in turn gives rise to the schemes of this paper, which are achieved by further in-place modifications of the original matrix.

Theorem 1.1. *[1] Let $A_{m \times n}$ be a nonzero matrix of rank r. Then there are a permutation matrix $P_{m \times m}$, a lower trapezoidal matrix $L_{m \times r}$ of rank r and an upper echelon matrix $U_{r \times n}$ of rank r such that $PA = LU$.*

The factorization of the theorem has numerous applications in the case that A has reduced rank. The compact scheme discussed here for A^+A is used extensively in the computations for [2]. The factorization itself can be used for the construction of constraints that define the image of a polyhedron under a linear transformation, also in [2]. The schemes we will detail involve further in-place (post factorization) preparation of the matrix A based on the fact that $A^+ = U^*(UU^*)^{-1}(L^*L)^{-1}L^*P$, as is easily shown using properties of the pseudoinverse.

The methods to be shown are based on modifications of the matrix R which we define as the upper left $r \times r$ submatrix of PA, and the submatrix of A corresponding to R. The diagonal of R consists of the pivot elements of the factor L. In the following we use the usual LDL^* variant of the Cholesky decomposition of a Hermitian positive definite matrix, expressed as such to avoid confusion with the factor L of the LU factorization. Sufficient detail is given for all of these schemes below that it is straightforward to convert any of them to a computer program.

1.1 A^+B

We first give the details for the calculation of A^+B as outlined in [1] which will then be applied to the other two cases. We assume that the matrix A has been factored according to Algorithm A.1 and that the integer arrays ρ and γ produced in the factorization have been retained. Here we assume that B is any $m \times p$ matrix.

Step 1. The process of calculating A^+B will overwrite portions of the factored form of A, so it is first necessary to calculate $C = L^*PB$. This is straightforward with, for $i = 0,...,r-1$ and $q = 0,...,p-1$,

$$C_{i,q} = \sum_{k=0}^{m-1} L_{k,i} B_{\rho_k,q} = \sum_{k=i}^{m-1} \bar{A}_{\rho_k,\gamma_i} B_{\rho_k,q} \qquad (1.1)$$

However, it is clear that these results can be stored column by column back into a set of r rows of each column of B. Therefore, rather than introducing a new matrix C, we shall overwrite the appropriate entries of B with the following assignments. Set

$$B_{\rho_i,q} \leftarrow \sum_{k=i}^{m-1} \bar{A}_{\rho_k,\gamma_i} B_{\rho_k,q} \qquad (1.2)$$

Volume - 4 Issue - 1

Citation: Stromberg, M. (2026). Compact Schemes for A^+B, A^+AB and AA^+B. *J Theor Phys Math*, *4*(1), 1-9.

Page 1 of 9

for $i = 0,...,r − 1$ and $q = 0,...,p − 1$. It is evident that entries to be overwritten are not needed in subsequent sums in (1.2) provided the given order is followed. The entries $B_{\rho i,q}$ will now contain L^*PB.

Step 2. Compute and store the lower triangular part of L^*L into A. Explicitly, the first r rows of L comprise an $r \times r$ lower triangular matrix in PA, namely the lower triangular part of R. We overwrite this matrix with the lower triangular part of L^*L by making the assignments

$$A_{\rho_j,\gamma_i} \leftarrow \sum_{k=j}^{m-1} A_{\rho_k,\gamma_j} \bar{A}_{\rho_k,\gamma_i} \qquad (1.3)$$

for $j = i,...,r − 1$ for $i = 0,...,r − 1$. Note that these assignments do not disturb the contents of U as stored in PA.

Step 3. Calculate $D = (L^*L)^{-1}C$. This is done by solving the system

$$L^*Ly' = y$$

for each column y' of D from each column y of C. Specifically, this step is application of the following algorithm. Again, rather than introducing a new matrix D, we overwrite the appropriate entries of B. This is an LDL* decomposition (consisting of (1.4), (1.5), and (1.6)) followed by forward substitution (1.7), application of the diagonal (1.8), and back substitution (1.9), which we state explicitly in terms of the factored contents of PA. Specifically, we find the LDL* factorization of L^*L from its lower triangular part, storing the factor L in the lower triangular $r \times r$ part of R with the diagonal D overwriting the pivot elements of L, since the diagonal elements of L have the value 1 and can be defined implicitly.

Algorithm HFS. *Let A be the result of the factorization of Algorithm A.1 and let ρ and γ be the integer arrays of length at least m and n respectively also resulting from the factorization. Assume that the r × r lower triangular part of R has been overwritten by the lower triangular part T of an r × r Hermitian positive definite matrix H (e.g., as in Step 2).*

$$
\begin{aligned}
&\text{for} \quad 0 \le i < r \quad \{ \\
&\quad \text{for} \quad 0 \le k < i \quad \{ \\
&\qquad A_{\rho_i,\gamma_i} \leftarrow A_{\rho_i,\gamma_i} - A_{\rho_i,\gamma_k} \bar{A}_{\rho_i,\gamma_k} A_{\rho_k,\gamma_k} \qquad (1.4) \\
&\quad \} \\
&\quad \text{for} \quad i+1 \le j < r \quad \{ \\
&\qquad \text{for} \quad 0 \le k < i \quad \{ \\
&\qquad\quad A_{\rho_j,\gamma_i} \leftarrow A_{\rho_j,\gamma_i} - A_{\rho_j,\gamma_k} \bar{A}_{\rho_i,\gamma_k} A_{\rho_k,\gamma_k} \qquad (1.5) \\
&\qquad \} \\
&\qquad A_{\rho_j,\gamma_i} \leftarrow A_{\rho_j,\gamma_i}/A_{\rho_i,\gamma_i} \qquad (1.6) \\
&\quad \} \\
&\} \\
&\text{for} \quad 0 \le q < p \quad \{ \\
&\quad \text{for} \quad 0 \le i < r \quad \{ \\
&\qquad \text{for} \quad 0 \le k < i \quad \{ \\
&\qquad\quad B_{\rho_i,q} \leftarrow B_{\rho_i,q} - A_{\rho_i,\gamma_k} B_{\rho_k,q} \qquad (1.7) \\
&\qquad \} \\
&\quad \} \\
&\quad \text{for} \quad 0 \le i < r \quad \{ \\
&\qquad B_{\rho_i,q} \leftarrow B_{\rho_i,q}/A_{\rho_i,\gamma_i} \qquad (1.8) \\
&\quad \} \\
&\quad \text{for} \quad r > i \ge 0 \quad \{ \\
&\qquad \text{for} \quad r > k > i \quad \{ \\
&\qquad\quad B_{\rho_i,q} \leftarrow B_{\rho_i,q} - \bar{A}_{\rho_k,\gamma_i} B_{\rho_k,q} \qquad (1.9) \\
&\qquad \} \\
&\quad \} \\
&\}
\end{aligned}
$$

Step 4. Compute and store the lower triangular part of $UU*$ into A, again overwriting the upper left $r \times r$ lower triangular matrix in PA. We overwrite this matrix with the lower triangular part of $UU*$ by making the assignments

$$A_{\rho_j, \gamma_i} \leftarrow \begin{cases} \bar{A}_{\rho_i, \gamma_j} & \text{if } j > i \\ 1 & \text{if } j = i \end{cases} + \sum_{k=1+\gamma_j}^{n-1} A_{\rho_j, k} \bar{A}_{\rho_i, k} \qquad (1.10)$$

for $j = i, \ldots, r-1$ for $i = 0, \ldots, r-1$.

Step 5. Calculate $F = (UU*)^{-1}D$ by again applying Algorithm HFS, where of course F now consists of the entries $B_{\rho_i, q}$ for $i = 0, \ldots, r-1$ and $q = 0, \ldots, p-1$.

Step 6. Complete the calculation of $A^+B = G = U*F$ with the assignments

$$G_{iq} = \sum_{k=0}^{r-1} B_{\rho_k, q} \begin{cases} 1 & \text{if } i = \gamma_k \\ \bar{A}_{\rho_k, i} & \text{if } i > \gamma_k \\ 0 & \text{if } i < \gamma_k \end{cases} \qquad (1.11)$$

for $i = 0, \ldots, n-1$ and $q = 0, \ldots, p-1$.

Calculation of the remaining quantities is for each split into two phases, namely preparation of the matrix A and then application of the result to the data B.

1.2 A^+AB

For this calculation we again assume that the matrix A has been factored according to Algorithm A.1 and that the integer arrays ρ and γ produced in the factorization have been retained. Here we assume that B is any $n \times p$ matrix. Steps 1 and 2 prepare the contents of A, the results of which are reusable with arbitrary B. Step 3 is application of A^+A to B.

Step 1. Preparation of A is continued from an initial LU factorization by first replacing the upper left $r \times r$ lower triangular matrix of PA with the lower triangular part of $UU*$ as in (1.10).

Step 2. Compute the LDL* decomposition of $UU*$ by applying only the factorization (1.4), (1.5), and (1.6) of Algorithm HFS. The calculation of A^+AB does not require retention of L of the factorization, so the scheme is free to overwrite that portion of the factored form of A.

Step 3. Use the following algorithm to apply A^+A to arbitrary data B. Simply put, this computes $U*(UU*)^{-1}UB$.

Algorithm *A^+AB. Let A be the result of the factorization of Algorithm A.1 and let ρ and γ be the integer arrays of length at least m and n respectively also resulting from the factorization. Assume that the upper left r × r lower triangular part of R has been overwritten by the lower triangular r × r factor T of the of the* LDL* *factored form of UU* and let B be an arbitrary n × p matrix. First calculate C = UB as follows in (1.12), then apply forward substitution (1.13), application of the diagonal (1.14) and back substitution (1.15) to obtain D = (UU*)⁻¹C, and finally A⁺AB = G = U*D (1.16).*

Again rather than introducing new matrices C and D, we overwrite the appropriate elements of B.

$$\text{for} \quad 0 \le q < p \quad \{$$
$$\quad \text{for} \quad 0 \le i < r \quad \{$$
$$\quad\quad B_{\gamma_i, q} \leftarrow B_{\gamma_i, q} + \sum_{j=\gamma_i+1}^{n-1} A_{\rho_i, j} B_{jq} \qquad (1.12)$$
$$\quad \}$$
$$\quad \text{for} \quad 0 \le i < r \quad \{$$
$$\quad\quad B_{\gamma_i, q} \leftarrow B_{\gamma_i, q} - \sum_{k=0}^{i-1} A_{\rho_i, \gamma_k} B_{\gamma_k, q} \qquad (1.13)$$
$$\quad \}$$
$$\quad \text{for} \quad 0 \le i < r \quad \{$$
$$\quad\quad B_{\gamma_i, q} \leftarrow B_{\gamma_i, q} / A_{\rho_i, \gamma_i} \qquad (1.14)$$
$$\quad \}$$

Citation: Stromberg, M. (2026). Compact Schemes for A⁺B, A⁺AB and AA⁺B. *J Theor Phys Math, 4*(1), 1-9.

Page 3 of 9

$$\text{for} \quad r > i \geq 0 \quad \{$$

$$B_{\gamma_i,q} \leftarrow B_{\gamma_i,q} - \sum_{k=i+1}^{r-1} \bar{A}_{\rho_k,\gamma_i} B_{\gamma_k,q} \qquad (1.15)$$

$$\}$$

$$\text{for} \quad 0 \leq i < n \quad \{$$

$$G_{iq} \leftarrow \sum_{k=0}^{r-1} B_{\gamma_k,q} \begin{cases} 1 & \text{if } i = \gamma_k \\ \bar{A}_{\rho_k,i} & \text{if } i > \gamma_k \\ 0 & \text{if } i < \gamma_k \end{cases} \qquad (1.16)$$

$$\}$$

$$\}$$

1.3   *AA⁺B*

We again assume that the matrix *A* has been factored according to Algorithm A.1 and that the integer arrays ρ and γ produced in the factorization have been retained. Here we assume that *B* is any $m \times p$ matrix. Steps 1–3 below prepare the contents of *A*, and again the results are reusable with arbitrary *B*. Step 4 is the application of *AA⁺* to *B*. First, note that

$$AA^+ = P^* L (L^* L)^{-1} L^* P \qquad (1.17)$$

and furthermore that (1.17) remains true if *L* is replaced by *LQ* where *Q* is any $r \times r$ invertible matrix. For the current application we will in particular take *Q* to be the diagonal of reciprocals of the pivot elements of *L*, namely the diagonal matrix with entries $\{1/A_{\rho_i,\gamma_i}\}$ for $i = 0,...,r-1$.

Step 1. Preparation of *A* is continued from an initial *LU* factorization by first replacing *L* with *LQ* in *PA* as follows.

$$\text{for} \quad 0 \leq i < r \quad \{$$

$$\quad \text{for} \quad i+1 \leq k < m \quad \{$$

$$\quad\quad A_{\rho_k,\gamma_i} \leftarrow A_{\rho_k,\gamma_i}/A_{\rho_i,\gamma_i} \qquad (1.18)$$

$$\quad \}$$

$$\}$$

Step 2. We will henceforth assume that *L* refers to the result of the replacements (1.18). Since we now can assume that the pivot elements of *L* (diagonal of *R*) all have the value 1, we can define these implicitly and are free to overwrite them, and thus to overwrite the upper triangular part of *R* without disturbing the contents of *L*. To this end we replace the upper triangular part of *R* with the upper triangular part of *L*L* as follows. Though this is stored as an upper triangular matrix, it can be treated as a lower triangular matrix for further applications, as explained below.

$$\text{for} \quad 0 \leq i < r \quad \{$$

$$\quad \text{for} \quad i \leq j < r \quad \{$$

$$\quad\quad A_{\rho_i,\gamma_j} \leftarrow \sum_{k=j}^{m-1} A'_{\rho_k,\gamma_i} \bar{A}'_{\rho_k,\gamma_j} \qquad (1.19)$$

$$\quad \}$$

$$\}$$

where we define

$$A'_{\rho_k,\gamma_q} = \begin{cases} A_{\rho_k,\gamma_q} & \text{if } k > q \\ 1 & \text{if } k = q \\ 0 & \text{if } k < q \end{cases} \qquad (1.20)$$

for $k = 0,\ldots,m = 1$ and $q = 0,\ldots,r-1$.

Step 3. The last step of preparation of *A* is to factor the matrix stored in Step 2 as though it is lower triangular and storing the transpose of the lower triangular factor into the upper triangular part of *R* as follows.

$$\text{for} \quad 0 \leq i < r \quad \{$$

$$\quad \text{for} \quad 0 \leq k < i \quad \{$$

$$\quad\quad A_{\rho_i,\gamma_i} \leftarrow A_{\rho_i,\gamma_i} - A_{\rho_k,\gamma_i} \bar{A}_{\rho_k,\gamma_i} A_{\rho_k,\gamma_k} \qquad (1.21)$$

$$
\begin{aligned}
&\}\\
&\text{for}\quad i+1 \le j < r \quad \{\\
&\quad\text{for}\quad 0 \le k < i\\
&\qquad A_{\rho_i,\gamma_j} \leftarrow A_{\rho_i,\gamma_j} - A_{\rho_k,\gamma_j}\bar{A}_{\rho_k,\gamma_i}A_{\rho_k,\gamma_k} \qquad\qquad (1.22)\\
&\quad\}\\
&\quad A_{\rho_i,\gamma_j} \leftarrow A_{\rho_i,\gamma_j}/A_{\rho_i,\gamma_i}\\
&\}\\
\}
\end{aligned}
$$

Step 4. Use the following algorithm to apply AA^+ to arbitrary data B. This computes $P_*L(L_*L)^{-1}L_*PB$, where L is the result of the replacements (1.18).

Algorithm *AA^+B. Let A be the result of the factorization of Algorithm A.1 and let ρ and γ be the integer arrays of length at least m and n respectively also resulting from the factorization. Assume that the $r \times r$ upper triangular part of R has been overwritten by the upper triangular $r \times r$ factor T from (1.21), (1.22) of the of the* LDL* *factored form of L_*L and let B be an arbitrary $m \times p$ matrix.*

First calculate $C = L_*PB$ as follows in (1.23), then apply forward substitution (1.24), application of the diagonal (1.25) and back substitution (1.26) to obtain $D = (L_*L)^{-1}C$, and finally $AA^+B = G = P_*LD$ (1.27). Again, we merely update the elements of B rather than introducing new matrices C and D.

$$
\begin{aligned}
&\text{for}\quad 0 \le q < p \quad \{\\
&\quad\text{for}\quad 0 \le i < r \quad \{\\
&\qquad B_{\rho_i,q} \leftarrow B_{\rho_i,q} + \sum_{k=i+1}^{m-1} \bar{A}_{\rho_k,\gamma_i}B_{\rho_k,q} \qquad\qquad (1.23)\\
&\quad\}\\
&\quad\text{for}\quad 0 \le i < r \quad \{\\
&\qquad B_{\rho_i,q} \leftarrow B_{\rho_i,q} - \sum_{k=0}^{i-1} A_{\rho_k,\gamma_i}B_{\rho_k,q} \qquad\qquad (1.24)\\
&\quad\}\\
&\quad\text{for}\quad 0 \le i < r \quad \{\\
&\qquad B_{\rho_i,q} \leftarrow B_{\rho_i,q}/A_{\rho_i,\gamma_i} \qquad\qquad\qquad\qquad (1.25)\\
&\quad\}\\
&\quad\text{for}\quad r > i \ge 0 \quad \{\\
&\qquad B_{\rho_i,q} \leftarrow B_{\rho_i,q} - \sum_{k=i+1}^{r-1} \bar{A}_{\rho_i,\gamma_k}B_{\rho_k,q} \qquad\qquad (1.26)\\
&\quad\}\\
&\quad\text{for}\quad 0 \le i < m \quad \{\\
&\qquad G_{\rho_i,q} \leftarrow \sum_{k=0}^{r-1} B_{\rho_k,q}
\begin{cases}
A_{\rho_i,\gamma_k} & \text{if } i > k\\
1 & \text{if } i = k\\
0 & \text{if } i < k
\end{cases} \qquad (1.27)\\
&\quad\}\\
&\}
\end{aligned}
$$

2. Example

We end this discussion by giving a simple example. Computations were done in double precision, with the results displayed here to 5 places. Starting with the rank 4 matrix

$$
A_{5x7} = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7\\
7 & 6 & 5 & 4 & 3 & 2 & 1\\
1 & 2 & 3 & 4 & 3 & 2 & 1\\
1 & 7 & 1 & 7 & 1 & 7 & 1\\
7 & 1 & 7 & 1 & 7 & 1 & 7
\end{bmatrix}
$$

we find its *LU* factorization according to Algorithm A.1, with

$$L = \begin{bmatrix} 7 & 0 & 0 & 0 \\ 1 & 6.14286 & 0 & 0 \\ 1 & 1.14286 & 2.23256 & 0 \\ 1 & 1.14286 & 2.23256 & 2 \\ 7 & -5 & 2.23256 & 2 \end{bmatrix}$$

and

$$U = \begin{bmatrix} 1 & 0.85714 & 0.71429 & 0.57143 & 0.42857 & 0.28571 & 0.14286 \\ 0 & 1 & 0.04651 & 1.04651 & 0.09302 & 1.09302 & 0.13953 \\ 0 & 0 & 1 & 1 & 1.10417 & 0.20833 & 0.31250 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 \end{bmatrix}$$

with row permutation $(0, 1, 2, 3, 4) \rightarrow (1, 3, 2, 0, 4)$. Applying the scheme of 1.1, we compute A^+B with B as the identity matrix I_{5x5}, which gives

$$A^+ = \begin{bmatrix} -0.02388 & 0.08326 & -0.15 & 0.01719 & 0.04219 \\ -0.01071 & 0.06071 & -0.10 & 0.05833 & -0.00833 \\ -0.03192 & 0.00379 & 0.15 & -0.04323 & 0.01510 \\ -0.01875 & -0.01875 & 0.20 & -0.00208 & -0.03542 \\ 0.00379 & -0.03192 & 0.15 & -0.04323 & 0.01510 \\ 0.06071 & -0.01071 & -0.10 & 0.05833 & -0.00833 \\ 0.08326 & -0.02388 & -0.15 & 0.01719 & 0.04219 \end{bmatrix}.$$

We next apply the scheme 1.2, computing A^+AA^+ taking $B = A^+$ and then scheme 1.3, computing AA^+A taking $B = A$ (the original A). The maximum difference between entries of A and AA^+A and between entries of A^+ and A^+AA^+ is found to be less than 2.7e-15. The results are identical whether using the fine, coarse or simple (with $\varepsilon = 1.0e-12$) methods from Appendix A.

A *LU* Factorization

The proof of Theorem 1.1 depends on a pair of lemmas that concern the construction of P, L and U in Algorithm A.1. The array ρ will contain the permutation information, so that $(PA)_{i,j} = A_{\rho_{i,j}}$. For the following we note that the changes to A (the assignments (A3), (A6)) take place only once for any position.

Algorithm A.1. *Let ρ and γ be integer arrays of length at least m and n respectively, and let D be a real array of length m. Let r and p be integers and set $r \leftarrow 0$ and $p \leftarrow 0$. For $0 \leq i < m$ set $\rho_i \leftarrow i$ and $D_i \leftarrow \|A_{i,\cdot}\|$ where $\|\,\|$ is a norm on the rows of A. Let R be an array of integers of length at least n.*

$$\text{for} \quad 0 \leq \ell < n \quad \{ \tag{A1}$$

$$\text{set} \quad C \leftarrow 0$$

$$\text{for} \quad r \leq i < m \quad \{ \tag{A2}$$

$$\text{if} \quad D_{\rho_i} > 0 \quad \{$$

$$A_{\rho_i,\ell} \leftarrow A_{\rho_i,\ell} - \sum_{0 \leq k < r} A_{\rho_i,\gamma_k} A_{\rho_k,\ell} \tag{A3}$$

$$\text{if} \quad C < |A_{\rho_i,\ell}|/D_{\rho_i} \quad \text{set} \quad C \leftarrow |A_{\rho_i,\ell}|/D_{\rho_i}, \quad p \leftarrow i$$

$$\}$$

$$\}$$

$$\text{if} \quad C > 0 \quad \{ \tag{A4}$$

$$\text{set} \quad \gamma_r \leftarrow \ell, \quad \rho_p \leftrightarrow \rho_r$$

$$\text{for} \quad \ell + 1 \leq j < n \quad \{ \tag{A5}$$

$$A_{\rho_r,j} \leftarrow (A_{\rho_r,j} - \sum_{0 \leq k < r} A_{\rho_r,\gamma_k} A_{\rho_k,j})/A_{\rho_r,\ell} \tag{A6}$$

$$\}$$

$$\text{set} \quad r \leftarrow r + 1$$

$$\}$$

Volume - 4 Issue - 1

Citation: Stromberg, M. (2026). Compact Schemes for *A+B, A+AB* and *AA+B. J Theor Phys Math, 4*(1), 1-9.   **Page 6 of 9**

$$\text{set} \quad R_\ell \leftarrow r - 1 \tag{A7}$$

$$\}$$

We define L and U in terms of the eventual value of A in Algorithm A.1 by

$$U_{p,q} = \begin{cases} 1 & \text{if } q = \gamma_p \\ A_{\rho_p,q} & \text{if } q > \gamma_p \\ 0 & \text{if } q < \gamma_p \end{cases} \tag{A8}$$

for $p = 0, ...r - 1$ and $q = 0, ..., n - 1$ and

$$L_{p,q} = \begin{cases} A_{\rho_p,\gamma_q} & \text{if } p \geq q \\ 0 & \text{if } p < q \end{cases} \tag{A9}$$

for $p = 0,...m - 1$ and $q = 0,...,r - 1$. These definitions are assumed to apply as soon as the assignments in (A3) and (A6) have taken place for the relevant entries of A.

Theorem 1.1 is established by examination of the algorithm and applying the following two lemmas. We assume the usual convention that vacuous sums are zero. By observation, the algorithm will proceed through the outer loop (A1), terminating after at most n steps. The vector R of the algorithm is a construct for use only in the lemmas, and can be omitted from implementations.

Lemma A.1. *[1] Let ℓ_0 be the smallest value of ℓ for which (A4) of Algorithm A.1 is satisfied. Then for each ℓ, $\ell_0 \leq \ell < n$, we have $0 \leq R_\ell < m$ and $\gamma_{R\ell} \leq \ell$. For any such ℓ let $\ell' = \ell + 1$. If $R_\ell = R_{\ell'}$ then $\gamma_{R\ell'} < \ell'$ and if $R_\ell < R_{\ell'}$ then $R_{\ell'} = R_\ell + 1$ and $\gamma R_{\ell'} = \ell'$. Furthermore either $\gamma R_\ell < \ell$, in which case we have*

$$(PA)_{i,\ell} = (LU)_{i,\ell} \tag{A10}$$

for $i = R_\ell + 1, \ldots, m - 1$, or $\gamma_{R_\ell} = \ell$ and we have

$$(PA)_{i,\ell} = (LU)_{i,\ell} \tag{A11}$$

for $i = R_\ell, \ldots, m - 1$ and

$$(PA)_{i,j} = (LU)_{i,j} \tag{A12}$$

for $i = R_\ell$ and $j = \ell + 1, \ldots, n - 1$. In any case,

$$(PA)_{i,j} = (LU)_{i,j} \tag{A13}$$

for $i = 0, \ldots, m - 1$ and $j = 0, \ldots, \ell_0 - 1$.

We give a slightly modified proof of the following.

Lemma A.2. *[1] For each $\ell = 0,...,n - 1$ we have*

$$(PA)_{i,q} = (LU)_{i,q} \tag{A14}$$

for $i = 0, \ldots, m - 1$ and $q = 0, \ldots, \ell$. If $\ell \geq \ell_0$ of Lemma A.1 then

$$(PA)_{p,j} = (LU)_{p,j} \tag{A15}$$

for $p = 0, \ldots, R_\ell$ and $j = \ell, \ldots, n - 1$.

Proof. We have (A14) for $\ell < \ell_0$ by Lemma A.1, so suppose $\ell \geq \ell_0$. Then $R_\ell \leq \ell$ and $\gamma R_\ell \leq \ell$, also by Lemma A.1 and its proof. Suppose the present lemma is true for some $\ell \geq 0$ and let $\ell' = \ell + 1$. If $R_\ell = R_{\ell'}$ then $\gamma R_{\ell'} < \ell'$. By the previous lemma, $(PA)_{i,\ell'} = (LU)_{i,\ell'}$ for $i = R_{\ell'} + 1, \ldots, m - 1$. By assumption we have $(PA)_{p,j} = (LU)_{p,j}$ for $p = 0, \ldots, R_\ell$ and $j = \ell, \ldots, n - 1$, which shows that $(PA)_{i,\ell'} = (LU)_{i,\ell'}$ for $i = 0, \ldots, R_{\ell'}$, thus $(PA)_{i,\ell'} = (LU)_{i,\ell'}$ for $i = 0, \ldots, m - 1$ and $(PA)_{p,j} = (LU)_{p,j}$ for $p = 0, \ldots, R_{\ell'}$ which is the induction hypothesis for ℓ'.

On the other hand if $R_\ell < R_{\ell'}$ then $R_{\ell'} = R_\ell + 1$ and $\gamma R_{\ell'} = \ell'$. Then $(PA)_{i,\ell'} = (LU)_{i,\ell'}$ for $i = R_{\ell'}, \ldots, m - 1$. Again by assumption we have $(PA)_{p,j} = (LU)_{p,j}$ for $p = 0, \ldots, R_\ell$ and $j = \ell, \ldots, n - 1$, which shows that $(PA)_{i,\ell'} = (LU)_{i,\ell'}$ for $i = 0, \ldots, R_{\ell'} - 1$ thus $(PA)_{i,\ell'} = (LU)_{i,\ell'}$ for $i = 0, \ldots, m - 1$. We have $(PA)_{p,j} = (LU)_{p,j}$ for $p = 0, \ldots, R_\ell$ and $j = \ell, \ldots, n - 1$ by assumption and $(PA)_{p,j} = (LU)_{p,j}$ for $p = R_{\ell'}$ and $j = \ell' + 1, \ldots, n - 1$ by Lemma A.1, and for $j = \ell'$ by choosing $i = R_{\ell'}$ above so that $(PA)_{p,j} = (LU)_{p,j}$ for $p = 0, \ldots, R_{\ell'}$ and $j = \ell', \ldots, n - 1$. This satisfies the inductive hypothesis. Now for ℓ_0, (A11) and (A12) hold, and we note that

$R_{\ell_0} = 0$. Then (A14) is true for $q = \ell_0$, and we have (A14) for $q < \ell_0$ by (A13), so (A14) holds for $i = 0, \ldots, m - 1$ and $q = 0, \ldots, \ell_0$. By (A12) we have (A15) for $j > \ell_0$. By (A11) we have (A15) for $j = \ell_0$. This instance completes the induction and proof of Lemma A.2.

Proof of Theorem 1.1 By Lemma A.2 it is clear that $PA = LU$. That L is lower trapezoidal (that is, a truncated lower triangular matrix) and U is upper echelon and that both have rank r follows by construction.

Note on machine arithmetic. Applications of the factorization will depend on use of machine numbers and in practice this means that the comparison (A4) can be misleading in the sense that a very small floating point number may either represent a value that is supposed to be zero or a nonzero value that just happens to be small. The value of C in the algorithm is the maximum of the nonzero values $|A_{\rho_i,\ell}|/D_{\rho_i'}$ if there is one, or is zero if not. Therefore, it is desirable to have a scheme for deciding which of these quantities is nonzero as opposed to being an artifact of machine arithmetic that appears nonzero. A number of methods can be used to separate these possibilities, one of which might be to simply find C as the maximum of the required machine values and to then stipulate that $C > \varepsilon$ (for some well-chosen ε) should be interpreted as that (A4) is in fact satisfied. This will be referred to as the *simple* method, and we will now discuss some adaptations of that approach. According to the results of [3], any order of evaluation of inner products returns a machine number $\hat{\nu}$ such that

$$|\hat{\nu} - x^T y| \leq nu|x|^T|y| \qquad (A16)$$

where x and y are vectors of length n and u is the machine unit.

Using (A16) it is easy to show that if ν_a is the machine-computed value of $|x|^T|y|$, and if

$$|\hat{\nu}| > \frac{nu}{1 - nu}\nu_a \qquad (A17)$$

then the exact arithmetic value of $x^T y$ is nonzero (where we assume that $nu < 1$). It should be noted that if (A16) is true, it is also true replacing n by K where K is the number of pairs (x_i, y_i) for which both components are nonzero, provided that the computation of $\hat{\nu}$ ignores pairs for which one component is zero (that is, the floating point representation of zero), so that (A16) and (A17) will at least sometimes be overestimates. In any case, for machine computations we find C by considering values of $|A_{\rho_i,\ell}|/D_{\rho_i}$ that are actually nonzero by using (A18), since by (A17) the exact arithmetic value of the right side of the assignment (A3) is nonzero if

$$|A_{\rho_i,\ell}| > \phi(K)\left(|A_{\rho_i,\ell}| + \sum_{0 \leq k < r} |A_{\rho_i,\gamma_k}||A_{\rho_k,\ell}|\right) \qquad (A18)$$

where we define $\phi(a) = au/(1 - au)$ and where K is determined in the process of evaluating the right side of (A18) as the count of nonzero terms in the sum, plus 1 if $A_{\rho_i,\ell} \neq 0$. The value to the left of the inequality in (A18) is after the assignment (A3) (i.e., the computed value), values to the right of the inequality are values before the assignment and all expressions in (A18) represent machine-computed values. Inequality (A18) applies for real computations. If x and y are complex, it suffices to consider the real and imaginary parts of x^*y separately. Let x_R and x_I represent the vectors of real and imaginary parts of x and define y_R and y_I similarly.

Then $\Re x^* y = x_R^T y_R + x_I^T y_I = (x_R, x_I)^T (y_R, y_I)$ where if a, b are vectors of length n, (a, b) represents the vector of length $2n$ consisting of the entries of a followed by the entries of b. It follows that if $\hat{\nu}$ is the machine value of x^*y then

$$|\Re(\hat{\nu} - x^*y)| \leq 2nu(|x_R|, |x_I|)^T(|y_R|, |y_I|) \qquad (A19)$$

from (A16) with similar results for imaginary parts. Letting $\tilde{\nu}_R$ and $\tilde{\nu}_I$ represent the machine computed values of $(|x_R|, |x_I|)^T(|y_R|, |y_I|)$ and $(|x_R|, |x_I|)^T(|y_I|, |y_R|)$ respectively, we have similarly to (A17) that if either

$$|\Re\hat{\nu}| > \phi(2n)\tilde{\nu}_R \quad \text{or} \quad |\Im\hat{\nu}| > \phi(2n)\tilde{\nu}_I \qquad (A20)$$

then $x^*y \neq 0$. The alternatives (A20) can be used to produce a pair of inequalities similar to (A18) for the complex case, namely

$$|\mathcal{R}_{\rho_i,\ell}| > \phi_{2r}\left(|\mathcal{R}_{\rho_i,\ell}| + \sum_{0 \leq k < r} |\mathcal{R}_{\rho_i,k}||\mathcal{R}_{\rho_k,\ell}| + |\mathcal{I}_{\rho_i,k}||\mathcal{I}_{\rho_k,\ell}|\right)$$
$$|\mathcal{I}_{\rho_i,\ell}| > \phi_{2r}\left(|\mathcal{I}_{\rho_i,\ell}| + \sum_{0 \leq k < r} |\mathcal{R}_{\rho_i,k}||\mathcal{I}_{\rho_k,\ell}| + |\mathcal{I}_{\rho_i,k}||\mathcal{R}_{\rho_k,\ell}|\right) \qquad (A21)$$

where $\phi_{2r} = \phi(2r + 1)$ and we define $R_{ij} = \Re A_{ij}$, and $I_{ij} = \Im A_{ij}$. A variation of (A21) can also be applied that counts nonzero terms in the sums. Application of (A18) or the variant (A21) (both of which will be called *fine* methods) will introduce extra floating-point operations. One approach to reduce the number of operations is to define μ as the

initial maximum value of $|A_{ij}|$ for $i = 0,...,m − 1$ and $j = 0,...,n − 1$ and μ_R, μ_I as $\max\limits_{i,j}|\Re A_{ij}|$ and $\max\limits_{i,j}|\Im A_{ij}|$ respectively, and to use (with $\kappa = \min\{m, n\}$)

$$|A_{\rho_i,\ell}| > \phi(\kappa + 1)(\mu + \kappa\mu^2) \qquad (A22)$$

or in the complex case

$$|\Re A_{\rho_i,\ell}| > \phi(2\kappa + 1)(\mu_R + \kappa\mu_R^2 + \kappa\mu_I^2)$$
$$|\Im A_{\rho_i,\ell}| > \phi(2\kappa + 1)(\mu_I + 2\kappa\mu_I\mu_R) \qquad (A23)$$

calculating the value of ϕ only once in either case and updating and retaining the entire right hand sides of the respective inequalities (A22) or (A23) only when μ, μ_R, or μ_I requires updating after (A3) or (A6). The inequalities (A22) or (A23) will be called *coarse* methods because they are overestimates compared to (A18) and (A21).

The simple method is typically the fastest, but requires a judicious choice of ε. Numerical experiments for real arithmetic indicate that on a modern processor on average and independent of the size of the matrix A, the fine method takes about 1.1 times as long as the simple method and the coarse method is between the simple and fine methods but very nearly equal to the simple method in execution time. Finally, it is possible for any of these methods to be wrong (especially the simple with $\varepsilon > 0$ or coarse methods, the fine method being somewhat more adaptive in this situation) by not concluding that $C > 0$, if A has many elements that are very small (within a few orders of magnitude of the machine unit) in absolute value. In that case, the algorithm will still produce a factorization, but the rank will be too small and the difference between PA and LU will effectively be those small elements of A. Whether or not this matters depends on the application. This situation might possibly be avoided by preparing A ahead of time by scaling or another transformation. One can also do this deliberately, forcing the situation by using a constant larger than the machine unit u in the above methods, in an attempt to filter 'noise' from A. At the other extreme, there is the simple method with $\varepsilon = 0$ i.e., the original algorithm, which treats every value not equal to floating point zero as nonzero [4].

References

1. Stromberg, M. (2020). LU factorization of any matrix. In New Sinc Methods of Numerical Analysis: Festschrift in Honor of Frank Stenger's 80th Birthday (pp. 369-382). Cham: Springer International Publishing.
2. Stromberg, M. (2023). A Class of Algorithms for Quadratic Minimization. arXiv preprint arXiv:2302.09139.
3. Jeannerod, C. P., & Rump, S. M. (2013). Improved error bounds for inner products in floating-point arithmetic. SIAM Journal on Matrix Analysis and Applications, 34(2), 338-344.
4. Stromberg, M. (2025). Compact Schemes for *A+B*, *A+AB* and *AA+B*. arXiv preprint arXiv:2511.12855v2.