Efficient recursive least squares solver for rank-deficient matrices

Ruben Staub, Stephan N. Steinmann

Efficient Recursive Least Squares Solver for Rank-Deficient Matrices

Ruben Staub

Univ Lyon, Ecole Normale Supérieure de Lyon, CNRS Université Lyon 1, Laboratoire de Chimie UMR 5182, 46 allée d'Italie, F-69364, LYON, France

Stephan N. Steinmann

Univ Lyon, Ecole Normale Supérieure de Lyon, CNRS Université Lyon 1, Laboratoire de Chimie UMR 5182, 46 allée d'Italie, F-69364, LYON, France

Abstract

Updating a linear least squares solution can be critical for near real-time signal-processing applications. The Greville algorithm proposes a simple formula for updating the pseudoinverse of a matrix $A \in \mathbb{R}^{n\times m}$ with rank r. In this paper, we explicitly derive a similar formula by maintaining a general rank factorization, which we call rank-Greville. Based on this formula, we implemented a recursive least squares algorithm exploiting the rank-deficiency of A, achieving the update of the minimum-norm least-squares solution in $O(mr)$ operations and, therefore, solving the linear least-squares problem from scratch in $O(nmr)$ operations. We empirically confirmed that this algorithm displays a better asymptotic time complexity than LAPACK solvers for rank-deficient matrices. The numerical stability of rank-Greville was found to be comparable to Cholesky-based solvers. Nonetheless, our implementation supports exact numerical representations of rationals, due to its remarkable algebraic simplicity.

Keywords: Moore-Penrose pseudoinverse, Generalized inverse, Recursive least squares, Rank-deficient linear systems

1. Introduction

In this paper, we are interested in computationally efficient algorithms for solving the recursive least-squares problem on rank-deficient matrices.

Let $\Gamma_i \in \mathbb{R}^m$ represent an observation of m variables, called regressors, associated with measurement y_i and let X_n be the unknown parameters of the linear relation:

$$A_n X_n + \epsilon_n = Y_n \tag{1}$$

Email addresses: ruben.staub@ens-lyon.org (Ruben Staub), stephan.steinmann@ens-lyon.fr (Stephan N. Steinmann)

where A_n is an $n \times m$ matrix representing n such observations, Y_n contains associated measurements and ϵ_n is the random disturbance:

$$A_n = \begin{pmatrix} {\Gamma_1}^\top \\ {\Gamma_2}^\top \\ \vdots \\ {\Gamma_n}^\top \end{pmatrix}, \quad Y_n = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{2}$$

The solution X_n to the general linear least-squares problem of equation 1 is defined as:

$$X_n = \arg\min_{x \in S}(||x||_2), \quad S = \arg\min_x(||A_n x - Y_n||_2) \tag{3}$$

This solution is unique[1], and sometimes called minimum-norm least-squares solution. Because of its uniqueness, it is sometimes simply referred to as the least-squares solution.

As demonstrated in the seminal paper by Penrose, the least-squares solution X_n can also be written[2]:

$$X_n = {A_n}^+ Y_n \tag{4}$$

where ${A_n}^+$ is the pseudoinverse of A_n, also called generalized inverse, or Moore-Penrose inverse. Due to its practical importance, the numerical determination of the generalized inverse remains an active topic of research.[3, 4]

The pseudoinverse $A^+ \in \mathbb{C}^{m \times n}$ of any matrix $A \in \mathbb{C}^{n \times m}$ is uniquely[5] characterized by the four Penrose equations[6]:

$$AA^+A = A \tag{5}$$

$$A^+AA^+ = A^+ \tag{6}$$

$$(AA^+)^\top = AA^+ \tag{7}$$

$$(A^+A)^\top = A^+A \tag{8}$$

Here, we are interested in a particular problem, i.e., updating a least-squares solution (or the generalized inverse) when a new observation (Γ_{n+1}, y_{n+1}) is added. This is typically called the recursive least-squares (RLS) problem for which the updated solution X_{n+1} is usually written[1, 7]:

$$X_{n+1} = X_n + K \times (y_{n+1} - {\Gamma_{n+1}}^\top X_n) \tag{9}$$

where $y_{n+1} - {\Gamma_{n+1}}^\top X_n$ is the predicted residual (or *a priori* error), and K is called the Kalman gain vector.

Algorithms that allow to update an already known previous solution can be of critical importance for embedded-systems signal processing, for example, as near real-time solutions might be required and new observations are added continuously[7]. Therefore, recursive least squares algorithms significantly benefit from the computational efficiency introduced by updating the least square solution instead of recomputing it from scratch.

If A_n has full column rank, the recursive least-squares solution X_{n+1} of equation (9) can be straightforwardly computed using normal equations[1]. This RLS algorithm has a time complexity of $O(m^2)$ for each update. Therefore, computing the solution for n successive observations, using equation (9), lead to a total time complexity of $O(nm^2)$.

However in the general case, A_n can be rank deficient, i.e. neither full column rank nor full row rank. This is indeed the case if we want to sample a large variable space (column deficiency), while accumulating data redundancy on the subspace of observed variables (row deficiency). Handling rank deficiency is, for example, desirable in neurocomputational learning applications[8]. Several algorithms have been developed to solve the rank-defficient recursive least-squares problem. In particular, the Greville algorithm[9] was designed for the recursive least-squares problem specifically, whereas most of the other algorithms are common least-squares solvers (based on Cholesky decomposition, QR factorization, SVD, ...) adapted to support updates of A_n (without the need to recompute the whole solution)[1].

The Greville algorithm provides an updated least squares solution, and additionally, an updated pseudoinverse at the same cost. This update step still has computational complexity in $O(m^2)$, independently of the rank deficiency of A_n. This leads to a $O(nm^2)$ time complexity for the computation of the full solution.[1] Variants of this algorithm were developed[10] based on the implicit decomposition of A_n, but still with an $O(m^2)$ update complexity[2].

In this paper, we write and implement a recursive least-squares algorithm that has single-update time complexity in $O(mr)$ (i.e. $O(nmr)$ total time complexity), where r is the rank of the matrix A_n. The underlying idea is to maintain a general rank decomposition into a full row rank matrix (purely underdetermined system) and a full column rank matrix (purely overdetermined system), which are much easier to treat in a recursive least squares procedure. Indeed, due to the rank deficiency of A_n these matrices have reduced sizes, leading to more efficient updates.

The remarkable simplicity of this approach makes it compatible with exact numerical representations in practice, without the need to use expensive symbolic computing. We also explore slightly more sophisticated rank decompositions, effectively bridging the gap between the Greville algorithm and QR-based recursive least-squares solvers.

---

[1] Note that one can easily reach $O(\max(n,m)\min(n,m)^2)$ using the property $({A_n}^\top)^+ = ({A_n}^+)^T$.

[2] Using the notations defined below, Albert and Sittler maintain the $m \times m$ matrices $(1 - {C_n}^\top \tilde{C}_n)$ and $({\tilde{C_n}}^\top {P_n}^{-1} \tilde{C}_n)$, whereas we maintain C_n, $\tilde{C}_n$ and ${P_n}^{-1}$ (whose sizes are reduced).

2. Derivation

2.1. General non-orthogonal rank factorization

Let A_n be a $n \times m$ matrix of rank r. A_n can be expressed as the product[9]:

$$A_n = B_n C_n \tag{10}$$

where B_n is a $n \times r$ matrix, and C_n is a $r \times m$ matrix, both of rank r, which corresponds to a full-rank factorization.

Let us consider a maximal free family $\mathcal{B}_\mathcal{A}$ of observations among observations $\Gamma_i \in \mathbb{R}^m$ in A_n. Note that $\mathcal{B}_\mathcal{A}$ is a basis of $\mathrm{Im}({A_n}^\top)$. Such basis is represented by the rows of C_n. Interpreting rows of C_n as observations $\Gamma_{C_i} \in \mathbb{R}^m$, we find that each observation Γ_{C_i} in C_n is linearly independent of the others. Hence C_n can then be seen as a purely underdetermined system. This system can be thought as linking the fitted value of each observation in $\mathcal{B}_\mathcal{A}$, to the fitted values for the m variables themselves.

Interpreting rows of B_n as observations $\gamma_i \in \mathbb{R}^r$, we find that each observation γ_i in B_n is the observation Γ_i of A_n expressed in the $\mathcal{B}_\mathcal{A}$ basis. Therefore, B_n can be seen as a purely overdetermined system, since each observation in $\mathcal{B}_\mathcal{A}$ is observed at least once. One can consider that this system links the value of each observation Γ_i in A_n, to the fitted values for each observation in $\mathcal{B}_\mathcal{A}$.

Theorem 1. *The pseudoinverse ${A_n}^+$ of A_n can be computed in $O(nmr)$ if matrices B_n and C_n verifying equation 10 are known. An explicit formula is then given by:*

$${A_n}^+ = {C_n}^\top ({C_n {C_n}^\top)^{-1} ({B_n}^\top B_n)^{-1} {B_n}^\top \tag{11}$$

Proof. By definition of B_n and C_n, ${B_n}^\top B_n$ and $C_n {C_n}^\top$ are both $r \times r$ matrices of rank r, and therefore non-singular. As a consequence, the explicit formula given is well defined as long as B_n and C_n correspond to a full-rank factorization.

It is straightforward to check that Eq. 11 satisfies all four Penrose equations (Eq. 5 to 8), and therefore, represents an acceptable pseudoinverse of A_n[9]. By the unicity of the Moore-Penrose inverse, we conclude that ${A_n}^+ = {C_n}^\top ({C_n {C_n}^\top)^{-1} ({B_n}^\top B_n)^{-1} {B_n}^\top$. Computing the pseudoinverse can, therefore, be reduced to computing the inverse of two $r \times r$ matrices and three matrix multiplications giving rise (in any order) to a total $O(nmr)$ time complexity, that could even be reduced by using faster algorithms[11]. □

We are now interested in the update of the pseudoinverse of A_n when adding a new observation Γ_{n+1}. Let us define A_{n+1} as:

$$A_{n+1} = \begin{pmatrix} A_n \\ {\Gamma_{n+1}}^\top \end{pmatrix} = B_{n+1} C_{n+1} \tag{12}$$

We distinguish two cases depending on the linear dependency of Γ_{n+1} with respect to previous observations $\Gamma_1, \ldots, \Gamma_n$. Note that we can equally well only consider the observations in $\mathcal{B}_\mathcal{A}$, since $\mathcal{B}_\mathcal{A}$ is a basis for $\mathrm{Vect}(\Gamma_1, \ldots, \Gamma_n)$.

Let $P_{\mathcal{B}_\mathcal{A}}$ be the projector into $\text{Vect}(\mathcal{B}_\mathcal{A}) = \text{Im}(C^\top)$. We define $\Gamma^p_{n+1} \in \mathbb{R}^m$ the projection of Γ_{n+1} into $\text{Vect}(\mathcal{B}_\mathcal{A})$. We also define $\gamma^p_{n+1} \in \mathbb{R}^r$ as Γ^p_{n+1} expressed in the $\mathcal{B}_\mathcal{A}$ basis. If $\mathcal{B}_\mathcal{A}$ was an orthonormal basis, the decomposition γ^p_{n+1} could be easily computed by inner products with $\gamma^p_{n+1} = C_n\Gamma_{n+1}$. However, in the general (non-orthonormal) case, the decomposition γ^p_{n+1} of $P_{\mathcal{B}_\mathcal{A}}\Gamma_{n+1}$ can be obtained using the dual $\tilde{\mathcal{B}_\mathcal{A}}$ of $\mathcal{B}_\mathcal{A}$, represented by the rows of $\tilde{C}_n$ defined as:

$$\begin{aligned}\tilde{C}_n &= (C_n {C_n}^\top)^{-1} C_n \\ &= \sigma_n^{-1} C_n\end{aligned} \tag{13}$$

where $\sigma_n = C_n {C_n}^\top$ is the Gram matrix of observations in $\mathcal{B}_\mathcal{A}$. γ^p_{n+1} can then be expressed by:

$$\begin{aligned}\gamma^p_{n+1} &= \tilde{C}_n P_{\mathcal{B}_\mathcal{A}} \Gamma_{n+1} \\ &= \tilde{C}_n \Gamma_{n+1}\end{aligned} \tag{14}$$

and Γ^p_{n+1} can then be expressed by:

$$\begin{aligned}\Gamma^p_{n+1} &= P_{\mathcal{B}_\mathcal{A}} \Gamma_{n+1} \\ &= {C_n}^\top \tilde{C}_n \Gamma_{n+1} \\ &= {C_n}^\top \gamma^p_{n+1}\end{aligned} \tag{15}$$

We define the rejection vector $\Gamma^{rej}_{n+1} \in \mathbb{R}^m$ associated with the projection of Γ_{n+1} into $\mathcal{B}_\mathcal{A}$:

$$\begin{aligned}\Gamma_{n+1} &= P_{\mathcal{B}_\mathcal{A}} \Gamma_{n+1} + \Gamma^{rej}_{n+1} \\ &= \Gamma^p_{n+1} + \Gamma^{rej}_{n+1} \\ &= {C_n}^\top \gamma^p_{n+1} + \Gamma^{rej}_{n+1} \\ &= {C_n}^\top \tilde{C}_n \Gamma_{n+1} + \Gamma^{rej}_{n+1}\end{aligned} \tag{16}$$

It becomes clear that Γ_{n+1} is linearly dependent from the previous observations $\Gamma_1, \ldots, \Gamma_n$, if and only if, Γ^{rej}_{n+1} is null. Note that γ^p_{n+1} , Γ^p_{n+1} and Γ^{rej}_{n+1} can be computed in $O(mr)$ if $\tilde{C}_n$ and C_n are already known.

The pseudoinverse ${A_n}^+$ can then be rewritten:

$${A_n}^+ = {\tilde{C}_n}^\top {P_n}^{-1} {B_n}^\top \tag{17}$$

where

$$P_n = {B_n}^\top B_n = \sum_{i=1}^{n} \gamma^p_i \cdot {\gamma^p_i}^\top \tag{18}$$

We finally define $\zeta_{n+1} \in \mathbb{R}^r$ and $\beta_{n+1} \in \mathbb{R}^n$ as:

$$\zeta_{n+1} = {P_n}^{-1} \gamma^p_{n+1}, \quad \beta_{n+1} = B_n \zeta_{n+1} \tag{19}$$

Note that if $\tilde{C}_n$, C_n and ${P_n}^{-1}$ are already known, ζ_{n+1} can be computed in $O(mr)$, but β_{n+1} can only be computed in $O(\max(m,n)r)$ if B_n is also known.

Theorem 2. *If $\Gamma_{n+1} \neq 0$ is linearly independent from previous observations, the pseudoinverse ${A_{n+1}}^+$ of A_{n+1} can be updated in $O(mn)$ if ${A_n}^+$, B_n, ${P_n}^{-1}$, C_n and $\tilde{C}_n$ are known. An explicit formula is then given by:*

$$ {A_{n+1}}^+ = \begin{pmatrix} {A_n}^+ & 0 \end{pmatrix} + \frac{\Gamma_{n+1}^{rej}}{\left\| \Gamma_{n+1}^{rej} \right\|_2^2} \begin{pmatrix} -{\beta_{n+1}}^\top & 1 \end{pmatrix} \tag{20} $$

Proof. First, let us observe how the full-rank factorization $A_{n+1} = B_{n+1}C_{n+1}$ is impacted by adding an observation Γ_{n+1} linearly independent from previous observations. Concerning C_{n+1}, we have:

$$ \begin{aligned} &\mathrm{Im}({C_{n+1}}^\top) = \mathrm{Im}({A_{n+1}}^\top) = \mathrm{Vect}(\mathcal{B}_\mathcal{A} \cup \{\Gamma_{n+1}\}) \neq \mathrm{Vect}(\mathcal{B}_\mathcal{A}) = \mathrm{Im}({A_n}^\top) = \mathrm{Im}({C_n}^\top) \\ &\Rightarrow \quad C_{n+1} \neq C_n \end{aligned} \tag{21} $$

Adding Γ_{n+1} to the rows of C_n leads to:

$$ C_{n+1} = \begin{pmatrix} C_n \\ {\Gamma_{n+1}}^\top \end{pmatrix}, \quad B_{n+1} = \begin{pmatrix} B_n & 0 \\ 0 & 1 \end{pmatrix} \tag{22} $$

It becomes clear from this definition that B_{n+1} has full column rank since B_n has full column rank, and also that C_{n+1} has full row rank. Therefore, B_{n+1} and C_{n+1} represent an acceptable full-rank decomposition of A_{n+1}, since we have:

$$ B_{n+1}C_{n+1} = \begin{pmatrix} B_n & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} C_n \\ {\Gamma_{n+1}}^\top \end{pmatrix} = \begin{pmatrix} B_nC_n \\ {\Gamma_{n+1}}^\top \end{pmatrix} = \begin{pmatrix} A_n \\ {\Gamma_{n+1}}^\top \end{pmatrix} = A_{n+1} \tag{23} $$

Second, we apply Theorem 1 to A_{n+1}:

$$ \begin{aligned} {A_{n+1}}^+ &= {C_{n+1}}^\top (C_{n+1} {C_{n+1}}^\top)^{-1} ({B_{n+1}}^\top B_{n+1})^{-1} {B_{n+1}}^\top \\ &= \begin{pmatrix} {C_n}^\top & \Gamma_{n+1} \end{pmatrix} \begin{pmatrix} C_n {C_n}^\top & C_n\Gamma_{n+1} \\ (C_n\Gamma_{n+1})^\top & {\Gamma_{n+1}}^\top \Gamma_{n+1} \end{pmatrix}^{-1} \begin{pmatrix} {B_n}^\top B_n & 0 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} {B_n}^\top & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} {C_n}^\top & \Gamma_{n+1} \end{pmatrix} \begin{pmatrix} C_n {C_n}^\top & C_n\Gamma_{n+1} \\ (C_n\Gamma_{n+1})^\top & {\Gamma_{n+1}}^\top \Gamma_{n+1} \end{pmatrix}^{-1} \begin{pmatrix} ({B_n}^\top B_n)^{-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} {B_n}^\top & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} {C_n}^\top & \Gamma_{n+1} \end{pmatrix} \begin{pmatrix} \sigma_n & C_n\Gamma_{n+1} \\ (C_n\Gamma_{n+1})^\top & {\Gamma_{n+1}}^\top \Gamma_{n+1} \end{pmatrix}^{-1} \begin{pmatrix} {P_n}^{-1} {B_n}^\top & 0 \\ 0 & 1 \end{pmatrix} \end{aligned} \tag{24} $$

Finally, we apply a generic block-wise inversion scheme:

$$ \begin{aligned} \begin{pmatrix} D & E \\ E^\top & F \end{pmatrix}^{-1} &= \begin{pmatrix} D^{-1} + D^{-1}E(F - E^\top D^{-1}E)^{-1}E^\top D^{-1} & -D^{-1}E(F - E^\top D^{-1}E)^{-1} \\ -(F - E^\top D^{-1}E)^{-1}E^\top D^{-1} & (F - E^\top D^{-1}E)^{-1} \end{pmatrix} \\ &= \begin{pmatrix} D^{-1} + D^{-1}ES^{-1}E^\top D^{-1} & -D^{-1}ES^{-1} \\ -S^{-1}E^\top D^{-1} & S^{-1} \end{pmatrix} \end{aligned} \tag{25} $$

where $S = F - E^\top D^{-1} E$, $D = \sigma_n$, $E = C_n \Gamma_{n+1}$ and $F = {\Gamma_{n+1}}^\top \Gamma_{n+1}$, since σ_n is non-singular and ${\Gamma_{n+1}}^\top \Gamma_{n+1}$ is not null since $\Gamma_{n+1} \neq 0$. This leads to the pseudoinverse formula:

$$
\begin{aligned}
{A_{n+1}}^+ &= \begin{pmatrix} {C_n}^\top & \Gamma_{n+1} \end{pmatrix} \begin{pmatrix} \sigma_n & C_n \Gamma_{n+1} \\ (C_n \Gamma_{n+1})^\top & {\Gamma_{n+1}}^\top \Gamma_{n+1} \end{pmatrix}^{-1} \begin{pmatrix} {P_n}^{-1} {B_n}^\top & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} {C_n}^\top & \Gamma_{n+1} \end{pmatrix} \begin{pmatrix} \sigma_n^{-1} + \sigma_n^{-1} C_n \Gamma_{n+1} S^{-1} (C_n \Gamma_{n+1})^\top \sigma_n^{-1} & -\sigma_n^{-1} C_n \Gamma_{n+1} S^{-1} \\ -S^{-1} (C_n \Gamma_{n+1})^\top \sigma_n^{-1} & S^{-1} \end{pmatrix} \begin{pmatrix} {P_n}^{-1} {B_n}^\top & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} {C_n}^\top & \Gamma_{n+1} \end{pmatrix} \begin{pmatrix} \sigma_n^{-1} + \gamma_{n+1}^p S^{-1} {\gamma_{n+1}^p}^\top & -S^{-1} \gamma_{n+1}^p \\ -S^{-1} {\gamma_{n+1}^p}^\top & S^{-1} \end{pmatrix} \begin{pmatrix} {P_n}^{-1} {B_n}^\top & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} {C_n}^\top & \Gamma_{n+1} \end{pmatrix} \left(\begin{pmatrix} \sigma_n^{-1} & 0 \\ 0 & 0 \end{pmatrix} + S^{-1} \begin{pmatrix} \gamma_{n+1}^p {\gamma_{n+1}^p}^\top & -\gamma_{n+1}^p \\ -{\gamma_{n+1}^p}^\top & 1 \end{pmatrix} \right) \begin{pmatrix} {P_n}^{-1} {B_n}^\top & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} {A_n}^+ & 0 \end{pmatrix} + S^{-1} \begin{pmatrix} {C_n}^\top & \Gamma_{n+1} \end{pmatrix} \begin{pmatrix} \gamma_{n+1}^p {\gamma_{n+1}^p}^\top & -\gamma_{n+1}^p \\ -{\gamma_{n+1}^p}^\top & 1 \end{pmatrix} \begin{pmatrix} {P_n}^{-1} {B_n}^\top & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} {A_n}^+ & 0 \end{pmatrix} + S^{-1} \begin{pmatrix} {C_n}^\top \gamma_{n+1}^p {\gamma_{n+1}^p}^\top {P_n}^{-1} {B_n}^\top - \Gamma_{n+1} \gamma_{n+1}^p {}^\top {P_n}^{-1} {B_n}^\top & -{C_n}^\top \gamma_{n+1}^p + \Gamma_{n+1} \end{pmatrix} \\
&= \begin{pmatrix} {A_n}^+ & 0 \end{pmatrix} + S^{-1} \begin{pmatrix} -\Gamma_{n+1}^{rej} {\beta_{n+1}}^\top & \Gamma_{n+1}^{rej} \end{pmatrix}
\end{aligned}
\tag{26}
$$

where the Schur complement S of σ_n is written as:

$$
\begin{aligned}
S &= {\Gamma_{n+1}}^\top \Gamma_{n+1} - {\Gamma_{n+1}}^\top {C_n}^\top \sigma_n^{-1} C_n \Gamma_{n+1} \\
&= {\Gamma_{n+1}}^\top \Gamma_{n+1} - {\Gamma_{n+1}}^\top \Gamma_{n+1}^p \\
&= {\Gamma_{n+1}}^\top \Gamma_{n+1}^{rej} \\
&= {\Gamma_{n+1}^{rej}}^\top \Gamma_{n+1}^{rej} = \left\| \Gamma_{n+1}^{rej} \right\|_2^2
\end{aligned}
\tag{27}
$$

S is, therefore, the square of the norm of the component of Γ_{n+1} along the orthogonal complement of $\mathrm{Im}({A_n}^\top)$. S is invertible since $\Gamma_{n+1}^{rej} \neq 0$.

Γ_{n+1}^{rej} and β_{n+1} can be computed in $O(\max(m,n)r)$ if C_n, $\tilde{C}_n$, B_n and ${P_n}^{-1}$ are already known. Therefore, the time complexity bottleneck is the outer product $\Gamma_{n+1}^{rej} {\beta_{n+1}}^\top$, leading to a total update in $O(mn)$. □

Theorem 3. *If Γ_{n+1} is a linear combination of previous observations, the pseudoinverse ${A_{n+1}}^+$ of A_{n+1} can be updated in $O(mn)$ if ${A_n}^+$, B_n, ${P_n}^{-1}$, C_n and $\tilde{C}_n$ are known. An explicit formula is then given by:*

$$
{A_{n+1}}^+ = \begin{pmatrix} {A_n}^+ & 0 \end{pmatrix} + \frac{{\tilde{C}_n}^\top \zeta_{n+1}}{1 + {\gamma_{n+1}^p}^\top \zeta_{n+1}} \begin{pmatrix} -{\beta_{n+1}}^\top & 1 \end{pmatrix} \tag{28}
$$

Proof. First, let us observe how the full-rank factorization $A_{n+1} = B_{n+1} C_{n+1}$ is impacted by adding an observation Γ_{n+1} that is a linear combination of previous

observations. Notice that:

$$\begin{aligned}\Gamma_{n+1} &= P_{\mathcal{B}_\mathcal{A}}\Gamma_{n+1} + \Gamma_{n+1}^{rej} = P_{\mathcal{B}_\mathcal{A}}\Gamma_{n+1} = \Gamma_{n+1}^{p} \\ &= {C_n}^\top \tilde{C}_n \Gamma_{n+1} \\ &= {C_n}^\top \gamma_{n+1}^{p}\end{aligned} \tag{29}$$

Since $\Gamma_{n+1} \in \text{Vect}(\Gamma_1, \ldots, \Gamma_n) = \text{Vect}(\mathcal{B}_\mathcal{A})$, $\mathcal{B}_\mathcal{A}$ is still a basis for $\text{Vect}(\Gamma_1, \ldots, \Gamma_n, \Gamma_{n+1})$. As a consequence, we can take $C_{n+1} = C_n$, leading to:

$$C_{n+1} = C_n, \quad B_{n+1} = \begin{pmatrix} B_n \\ {\gamma_{n+1}^{p}}^\top \end{pmatrix} \tag{30}$$

From this definition follows that C_{n+1} still has full row rank, and also that B_{n+1} has full column rank since B_n has full column rank. Therefore, B_{n+1} and C_{n+1} represent an acceptable full-rank decomposition of A_{n+1}, since we have:

$$B_{n+1}C_{n+1} = \begin{pmatrix} B_n \\ {\gamma_{n+1}^{p}}^\top \end{pmatrix} C_n = \begin{pmatrix} B_n C_n \\ {\gamma_{n+1}^{p}}^\top C_n \end{pmatrix} = \begin{pmatrix} A_n \\ {\Gamma_{n+1}}^\top \end{pmatrix} = A_{n+1} \tag{31}$$

Second, we apply Theorem 1 to A_{n+1}:

$$\begin{aligned}{A_{n+1}}^+ &= {C_{n+1}}^\top (C_{n+1}{C_{n+1}}^\top)^{-1} (B_{n+1}{}^\top B_{n+1})^{-1} {B_{n+1}}^\top \\ &= {C_n}^\top (C_n {C_n}^\top)^{-1} \left(\begin{pmatrix} {B_n}^\top & \gamma_{n+1}^{p} \end{pmatrix} \begin{pmatrix} B_n \\ {\gamma_{n+1}^{p}}^\top \end{pmatrix} \right)^{-1} \begin{pmatrix} {B_n}^\top & \gamma_{n+1}^{p} \end{pmatrix} \\ &= {C_n}^\top (C_n {C_n}^\top)^{-1} \left({B_n}^\top B_n + \gamma_{n+1}^{p} {\gamma_{n+1}^{p}}^\top \right)^{-1} \begin{pmatrix} {B_n}^\top & \gamma_{n+1}^{p} \end{pmatrix} \\ &= {\tilde{C}_n}^\top \left(P_n + \gamma_{n+1}^{p} {\gamma_{n+1}^{p}}^\top \right)^{-1} \begin{pmatrix} {B_n}^\top & \gamma_{n+1}^{p} \end{pmatrix}\end{aligned} \tag{32}$$

Finally, we apply the Sherman-Morrison formula stating that for any non-singular matrix $G \in \mathbb{R}^{n \times n}$ and any vector $v \in \mathbb{R}^n$, if $G + vv^\top$ is non-singular, then:

$$\left(G + vv^\top\right)^{-1} = G^{-1} - \frac{G^{-1}vv^\top G^{-1}}{1 + v^\top G^{-1} v} \tag{33}$$

with $G = P_n$ and $v = \gamma_{n+1}^{p}$, since P_n and $P_{n+1} = B_{n+1}{B_{n+1}}^\top$ are non-singular[3].

[3] $P_{n+1} = B_{n+1}{B_{n+1}}^\top$ is non-singular, since P_{n+1} is square with full rank. Indeed, $\text{rank}(P_{n+1}) = \text{rank}(B_{n+1}) = r$, using theorem 5.5.4 of [12]

This leads to the pseudoinverse formula:

$$
\begin{aligned}
{A_{n+1}}^{+} &= {\tilde{C}_n}^{\top}\left(P_n + \gamma^p_{n+1}{\gamma^p_{n+1}}^{\top}\right)^{-1}\begin{pmatrix}{B_n}^{\top} & \gamma^p_{n+1}\end{pmatrix} \\
&= {\tilde{C}_n}^{\top}\left({P_n}^{-1} - \frac{{P_n}^{-1}\gamma^p_{n+1}{\gamma^p_{n+1}}^{\top}{P_n}^{-1}}{1+{\gamma^p_{n+1}}^{\top}{P_n}^{-1}\gamma^p_{n+1}}\right)\begin{pmatrix}{B_n}^{\top} & \gamma^p_{n+1}\end{pmatrix} \\
&= {\tilde{C}_n}^{\top}\left({P_n}^{-1} - \frac{\zeta_{n+1}{\zeta_{n+1}}^{\top}}{1+{\gamma^p_{n+1}}^{\top}\zeta_{n+1}}\right)\begin{pmatrix}{B_n}^{\top} & \gamma^p_{n+1}\end{pmatrix} \\
&= \begin{pmatrix}{A_n}^{+} & {\tilde{C}_n}^{\top}\zeta_{n+1}\end{pmatrix} - {\tilde{C}_n}^{\top}\frac{\zeta_{n+1}{\zeta_{n+1}}^{\top}}{1+{\gamma^p_{n+1}}^{\top}\zeta_{n+1}}\begin{pmatrix}{B_n}^{\top} & \gamma^p_{n+1}\end{pmatrix} \\
&= \begin{pmatrix}{A_n}^{+} & {\tilde{C}_n}^{\top}\zeta_{n+1}\end{pmatrix} - \frac{{\tilde{C}_n}^{\top}\zeta_{n+1}}{1+{\gamma^p_{n+1}}^{\top}\zeta_{n+1}}\begin{pmatrix}{\beta_{n+1}}^{\top} & {\zeta_{n+1}}^{\top}\gamma^p_{n+1}\end{pmatrix} \\
&= \begin{pmatrix}{A_n}^{+} & 0\end{pmatrix} + \frac{{\tilde{C}_n}^{\top}\zeta_{n+1}}{1+{\gamma^p_{n+1}}^{\top}\zeta_{n+1}}\begin{pmatrix}-{\beta_{n+1}}^{\top} & 1+{\gamma^p_{n+1}}^{\top}\zeta_{n+1} - {\zeta_{n+1}}^{\top}\gamma^p_{n+1}\end{pmatrix} \\
&= \begin{pmatrix}{A_n}^{+} & 0\end{pmatrix} + \frac{{\tilde{C}_n}^{\top}\zeta_{n+1}}{1+{\gamma^p_{n+1}}^{\top}\zeta_{n+1}}\begin{pmatrix}-{\beta_{n+1}}^{\top} & 1\end{pmatrix}
\end{aligned}
\tag{34}
$$

γ^p_{n+1} , ζ_{n+1} and β_{n+1} can be computed in $O(\max(m,n)r)$ if $\tilde{C}_n$, B_n and ${P_n}^{-1}$ are already known. Therefore, the time complexity bottleneck is the outer product $\left({\tilde{C}_n}^{\top}\zeta_{n+1}\right){\beta_{n+1}}^{\top}$, leading to a total update in $O(mn)$. □

Corollary 3.1. *For any observation $\Gamma_{n+1} \in \mathbb{R}^m$, the pseudoinverse ${A_{n+1}}^{+}$ of A_{n+1} can be updated in $\Theta(mn)$ if ${A_n}^{+}$, B_n, ${P_n}^{-1}$, C_n and $\tilde{C}_n$ are known.*

Indeed, at least $n \times m$ terms of the pseudoinverse need to be updated when adding a new observation, in the general case[4]. Therefore, the pseudoinverse update has a fundamental cost component that cannot be improved, hence the $\Theta(mn)$ complexity. This limitation is not present in the recursive least square problem. In this problem, we are only interested in updating the least square solution X_{n+1} when adding a new observation Γ_{n+1} with associated target y_{n+1}:

$$
X_{n+1} = \begin{pmatrix}A_n \\ {\Gamma_{n+1}}^{\top}\end{pmatrix}^{+}\begin{pmatrix}Y_n \\ y_{n+1}\end{pmatrix} = {A_{n+1}}^{+}\begin{pmatrix}Y_n \\ y_{n+1}\end{pmatrix} \tag{35}
$$

Theorem 4. *If $\Gamma_{n+1} \neq 0$ is linearly independent from previous observations, the least square solution X_{n+1} can be updated in $O(mr)$ if X_n, C_n and $\tilde{C}_n$ are*

[4]To be convinced, consider $A_n = I_n$ the identity matrix and $\Gamma_{n+1} = \begin{pmatrix}1 \\ \vdots \\ 1\end{pmatrix}$.

known. An explicit formula (in the form of equation 9) is then given by:

$$X_{n+1} = X_n + \frac{\Gamma_{n+1}^{rej}}{\left\|\Gamma_{n+1}^{rej}\right\|_2^2}\left(y_{n+1} - {\Gamma_{n+1}}^\top X_n\right) \tag{36}$$

Proof. First, let us inject theorem 2 into the definition of X_{n+1}:

$$\begin{aligned} X_{n+1} &= {A_{n+1}}^+ \begin{pmatrix} Y_n \\ y_{n+1} \end{pmatrix} \\ &= {A_n}^+ Y_n + \frac{\Gamma_{n+1}^{rej}}{\left\|\Gamma_{n+1}^{rej}\right\|_2^2}\begin{pmatrix} -{\beta_{n+1}}^\top & 1 \end{pmatrix}\begin{pmatrix} Y_n \\ y_{n+1} \end{pmatrix} \\ &= X_n + \frac{\Gamma_{n+1}^{rej}}{\left\|\Gamma_{n+1}^{rej}\right\|_2^2}\left(y_{n+1} - {\beta_{n+1}}^\top Y_n\right) \end{aligned} \tag{37}$$

Let us simplify further this equation by recognizing ${\beta_{n+1}}^\top Y_n$ as the fitted target associated with Γ_{n+1}:

$$\begin{aligned} X_{n+1} &= X_n + \frac{\Gamma_{n+1}^{rej}}{\left\|\Gamma_{n+1}^{rej}\right\|_2^2}\left(y_{n+1} - {\Gamma_{n+1}}^\top {\tilde{C}_n}^\top {P_n}^{-1} {B_n}^\top Y_n\right) \\ &= X_n + \frac{\Gamma_{n+1}^{rej}}{\left\|\Gamma_{n+1}^{rej}\right\|_2^2}\left(y_{n+1} - {\Gamma_{n+1}}^\top {A_n}^+ Y_n\right) \\ &= X_n + \frac{\Gamma_{n+1}^{rej}}{\left\|\Gamma_{n+1}^{rej}\right\|_2^2}\left(y_{n+1} - {\Gamma_{n+1}}^\top X_n\right) \\ &= X_n + \frac{\Gamma_{n+1}^{rej}}{\left\|\Gamma_{n+1}^{rej}\right\|_2^2}\,\Delta y_{n+1} \end{aligned} \tag{38}$$

where $\Delta y_{n+1} = y_{n+1} - {\Gamma_{n+1}}^\top X_n$ is the difference between the expected/fitted target (i.e. $\Gamma_{n+1}{X_n}^\top$) and the real target y_{n+1} associated with the new observation Γ_{n+1} (i.e. the predicted residual, or *a priori* error). We identify $\frac{\Gamma_{n+1}^{rej}}{\left\|\Gamma_{n+1}^{rej}\right\|_2^2}$ to be the associated Kalman gain vector in this case[1].

Γ_{n+1}^{rej} can be computed in $O(mr)$ if C_n and $\tilde{C}_n$ are already known, which is the time complexity bottleneck of the whole update step. □

Theorem 5. *If Γ_{n+1} is a linear combination of previous observations, the least square solution X_{n+1} can be updated in $O(mr)$ if X_n, C_n, $\tilde{C}_n$ and ${P_n}^{-1}$ are known. An explicit formula (in the form of equation 9) is then given by:*

$$X_{n+1} = X_n + \frac{{\tilde{C}_n}^\top \zeta_{n+1}}{1 + {\gamma_{n+1}^p}^\top \zeta_{n+1}}\left(y_{n+1} - {\Gamma_{n+1}}^\top X_n\right) \tag{39}$$

Proof. Let us proceed similarly to theorem 4, by injecting theorem 3 into the definition of X_{n+1}:

$$\begin{aligned} X_{n+1} &= {A_{n+1}}^{+} \begin{pmatrix} Y_n \\ y_{n+1} \end{pmatrix} \\ &= {A_n}^{+} Y_n + \frac{{\tilde{C}_n}^{\top} \zeta_{n+1}}{1 + {\gamma_{n+1}^p}^{\top} \zeta_{n+1}} \begin{pmatrix} -{\beta_{n+1}}^{\top} & 1 \end{pmatrix} \begin{pmatrix} Y_n \\ y_{n+1} \end{pmatrix} \\ &= X_n + \frac{{\tilde{C}_n}^{\top} \zeta_{n+1}}{1 + {\gamma_{n+1}^p}^{\top} \zeta_{n+1}} \left(y_{n+1} - {\beta_{n+1}}^{\top} Y_n \right) \\ &= X_n + \frac{{\tilde{C}_n}^{\top} \zeta_{n+1}}{1 + {\gamma_{n+1}^p}^{\top} \zeta_{n+1}} \left(y_{n+1} - {\Gamma_{n+1}}^{\top} X_n \right) \\ &= X_n + \frac{{\tilde{C}_n}^{\top} \zeta_{n+1}}{1 + {\gamma_{n+1}^p}^{\top} \zeta_{n+1}} \Delta y_{n+1} \end{aligned} \tag{40}$$

where $\Delta y_{n+1} = y_{n+1} - {\Gamma_{n+1}} {X_n}^{\top}$ (i.e. the predicted residual, or *a priori* error). We identify $\frac{{\tilde{C}_n}^{\top} \zeta_{n+1}}{1 + {\gamma_{n+1}^p}^{\top} \zeta_{n+1}}$ to be the associated Kalman gain vector in this case.

γ_{n+1}^p and ζ_{n+1} can be computed in $O(mr)$ if C_n, $\tilde{C}_n$ and ${P_n}^{-1}$ are already known. The whole update step can then be performed in $O(mr)$ operations. □

Theorem 6. *For any new observation $\Gamma_{n+1} \in \mathbb{R}^m$, the matrices C_{n+1}, $\tilde{C}_{n+1}$ and ${P_{n+1}}^{-1}$ can be updated in $O(mr)$ if C_n, $\tilde{C}_n$ and ${P_n}^{-1}$ are already known.*

Proof. The updating formula naturally depends on the linear dependency of Γ_{n+1} from previous observations (i.e. whether Γ_{n+1}^{rej} is non-null). Let us note that Γ_{n+1}^{rej} itself can be computed in $O(mr)$ operations if $\tilde{C}_n$ and C_n are known.

If Γ_{n+1} is linearly independent from previous observations (i.e. $\Gamma_{n+1}^{rej} \neq 0$), equation 22 is valid, leading to:

$$\begin{aligned} P_{n+1}^{-1} &= \left({B_{n+1}}^{\top} B_{n+1} \right)^{-1} = \left(\begin{pmatrix} {B_n}^{\top} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} B_n & 0 \\ 0 & 1 \end{pmatrix} \right)^{-1} = \begin{pmatrix} {B_n}^{\top} B_n & 0 \\ 0 & 1 \end{pmatrix}^{-1} \\ &= \begin{pmatrix} {P_n}^{-1} & 0 \\ 0 & 1 \end{pmatrix} \end{aligned} \tag{41}$$

$$C_{n+1} = \begin{pmatrix} C_n \\ {\Gamma_{n+1}}^{\top} \end{pmatrix} \tag{42}$$

Using equation 26, we can write:

$$\tilde{C}_{n+1} = (C_{n+1}{C_{n+1}}^\top)^{-1}C_{n+1} = \begin{pmatrix} \sigma_n^{-1} + \frac{\gamma^p_{n+1}{\gamma^p_{n+1}}^\top}{\left\|\Gamma^{rej}_{n+1}\right\|_2^2} & -\frac{\gamma^p_{n+1}}{\left\|\Gamma^{rej}_{n+1}\right\|_2^2} \\ -\frac{{\gamma^p_{n+1}}^\top}{\left\|\Gamma^{rej}_{n+1}\right\|_2^2} & \frac{1}{\left\|\Gamma^{rej}_{n+1}\right\|_2^2} \end{pmatrix} \begin{pmatrix} C_n \\ \Gamma_{n+1} \end{pmatrix}$$
$$= \begin{pmatrix} \tilde{C}_n - \gamma^p_{n+1}\frac{{\Gamma^{rej}_{n+1}}^\top}{\left\|\Gamma^{rej}_{n+1}\right\|_2^2} \\ \frac{{\Gamma^{rej}_{n+1}}^\top}{\left\|\Gamma^{rej}_{n+1}\right\|_2^2} \end{pmatrix} \tag{43}$$

These formulae can be applied in $O(mr)$ operations, since γ^p_{n+1} and Γ^{rej}_{n+1} can themselves be computed in $O(mr)$ operations if $\tilde{C}_n$ and C_n are already known.

If Γ_{n+1} is a linear combination of previous observations (i.e. $\Gamma^{rej}_{n+1} = 0$), equation 30 is valid, leading to:

$$C_{n+1} = C_n, \quad \tilde{C}_{n+1} = \tilde{C}_n \tag{44}$$

Using equation 34, we can write:

$${P_{n+1}}^{-1} = {P_n}^{-1} - \frac{\zeta_{n+1}{\zeta_{n+1}}^\top}{1 + {\gamma^p_{n+1}}^\top\zeta_{n+1}} \tag{45}$$

This formula can be applied in $O(mr)$ operations[5], since γ^p_{n+1} and ζ_{n+1} can themselves be computed in $O(mr)$ operations if ${P_n}^{-1}$, $\tilde{C}_n$ and C_n are already known.

□

Corollary 6.1. *For any $n \times m$ matrix A_n of rank r and any vector $Y_n \in \mathbb{R}^n$, the least square solution $X_n = {A_n}^+ Y_n$ can be computed in $O(mnr)$ operations.*

2.2. Orthogonal rank factorization

The theorems regarding the update of the pseudo-inverse ${A_{n+1}}^+$ (theorems 2 and 3) and least-squares solution X_{n+1} (theorems 4 and 5) are valid for any decomposition satisfying equation 10. Therefore, the rows of C_n can be required to form an orthogonal basis. This can be easily achieved by storing only the rejection vectors Γ^{rej}_{n+1} into C_{n+1}. More generally, one can store rescaled rejection vectors $\alpha_{n+1}\Gamma^{rej}_{n+1}$ instead, with $0 \neq \alpha_{n+1} \in \mathbb{R}$. Theorem 6 remains valid, with equations 41, 42 and 43 becoming:

$$C_{n+1} = \begin{pmatrix} C_n \\ \alpha_{n+1}{\Gamma^{rej}_{n+1}}^\top \end{pmatrix} \tag{46}$$

[5]Note that this complexity reduces to $O(r^2)$ if γ^p_{n+1} is already known.

$$\tilde{C}_{n+1} = \begin{pmatrix} \tilde{C}_n \\ \frac{{\Gamma_{n+1}^{rej}}^\top}{\alpha_{n+1}\left\|\Gamma_{n+1}^{rej}\right\|_2^2} \end{pmatrix} \tag{47}$$

$$\begin{aligned} P_{n+1}^{-1} &= \left({B_{n+1}}^\top B_{n+1}\right)^{-1} \\ &= \left(\begin{pmatrix} {B_n}^\top & \gamma_{n+1}^p \\ 0 & \frac{1}{\alpha_{n+1}} \end{pmatrix}\begin{pmatrix} B_n & 0 \\ {\gamma_{n+1}^p}^\top & \frac{1}{\alpha_{n+1}} \end{pmatrix}\right)^{-1} \\ &= \begin{pmatrix} {P_n}^{-1} & -\alpha_{n+1}\zeta_{n+1} \\ -\alpha_{n+1}{\zeta_{n+1}}^\top & \alpha_{n+1}^2(1+{\gamma_{n+1}^p}^\top\zeta_{n+1}) \end{pmatrix} \end{aligned} \tag{48}$$

In particular, one can consider $\alpha_{n+1} = \left\|\Gamma_{n+1}^{rej}\right\|_2^{-1}$. In this case the rows of C_n form an orthonormal basis, i.e., C_n is an orthogonal matrix (i.e. $\tilde{C}_n = C_n$), and equation 10 becomes a Gram-Schmidt based thin LQ decomposition. This variant offers slightly reduced storage and update computational time.

3. Implementation

Based on the theorems above, one can devise a simple algorithm satisfying corollary 6.1, the pseudocode of which is shown in Algorithm 1.

Algorithm 1 Rank-deficient RLS, also called rank-Greville

procedure UPDATELEASTSQUARES(Γ, y, X, C, $\tilde{C}$, P^{-1})
  $\gamma \leftarrow \tilde{C}\,\Gamma$
  $\Gamma_r \leftarrow \Gamma - C^\top\gamma$
  if $\Gamma_r \neq 0$ **then**
    $K \leftarrow \frac{\Gamma_r}{\|\Gamma_r\|_2^2}$
    $C \leftarrow \begin{pmatrix} C \\ \Gamma^\top \end{pmatrix}$
    $\tilde{C} \leftarrow \begin{pmatrix} \tilde{C} - \gamma\,K^\top \\ K^\top \end{pmatrix}$
    $P^{-1} \leftarrow \begin{pmatrix} P^{-1} & 0 \\ 0 & 1 \end{pmatrix}$
  else
    $\zeta \leftarrow P^{-1}\gamma$
    $K \leftarrow \frac{\tilde{C}^\top\zeta}{1+\gamma^\top\zeta}$
    $P^{-1} \leftarrow P^{-1} - \frac{\zeta\,\zeta^\top}{1+\gamma^\top\zeta}$
  end if
  $X \leftarrow X + K \times \left(y - \Gamma^\top X\right)$
end procedure

These formulae have been implemented in Python3 using the Numpy library. In addition to the least-squares update algorithm (in $O(mr)$ operations),

this implementation also supports pseudo-inverse (in $O(mn)$ operations) and covariance matrix[13] updates (in $O(m^2)$ operations).

The orthogonal and orthonormal basis variants described in section 2.2 have also been implemented for least-squares update (in $O(mr)$ operations) and pseudo-inverse update (in $O(mn)$ operations).

In practice, checking if $\Gamma_{n+1}^{rej} \neq 0$ is ill-defined with floating-point arithmetic. Yet, it is crucial in this algorithm as it determines the effective rank of the linear system. Therefore, we define a threshold *eps* so that Γ_{n+1} is considered a linear combination of previous observations if and only if:

$$||\Gamma_{n+1}^{rej}||_2 < eps \quad \text{or} \quad ||\Gamma_{n+1}^{rej}||_2 < eps \times ||\Gamma_{n+1}||_2 \tag{49}$$

By default, *eps* is set to $(m^2r + mr + m) \times \epsilon_M$ in order to account for rounding error propagation, with ϵ_M being the machine precision.

It is important to note that the general non-orthogonal basis implementation does support exact representations such as those defined in Python's *fractions* module. Indeed, this scheme (as the Greville algorithm) uses only operations well defined on rational numbers. One should note that the orthogonal basis variant scheme is also compatible with exact numerical representations as long as the rescaling factors α_{n+1} can themselves be represented exactly.

4. Numerical Tests

All computations were performed using Python 3.6.9[14, 15] with Scipy version 1.4.1[16] and Numpy version 1.18.3[17, 18] linked with OpenBLAS on an Intel Xeon W-2123 CPU with DDR4-2666 RAM. The code used for the numerical tests is available along with our Python3 implementation in the supporting information and will soon become available on `https://github.com/RubenStaub/rank-greville`.

4.1. Computational efficiency

In this section we empirically evaluate the computational efficiency achieved by the rank factorization Greville algorithm described in this paper (Algorithm 1), referred to as "rank-Greville". Note that for comparison purposes, its total complexity (i.e. computing the full least-squares solution from scratch in $O(nmr)$ operations) is studied here, even though this algorithm was not specifically designed as a least-squares solver from scratch[6]. Rather, rank-Greville was designed for the recursive update in applications requiring a constantly up-to-date solution.

For this analysis to be meaningful, the rank-Greville algorithm time efficiency is compared against standard algorithms from the LAPACK library:

[6]Indeed, in this case rank-Greville must perform much more memory allocation and copy than a conventional solver, explaining, in part, the large prefactor

- "gelsy" refers to the DGELSY least-squares solver from the LAPACK library based on a complete orthogonal factorization (using QR factorization with column pivoting)[19, 20].
- "gelss" refers to the DGELSS least-squares solver from the LAPACK library based on SVD[19].
- "gelsd" refers to the DGELSD least-squares solver from the LAPACK library also based on SVD, using a diagonal form reduction[19].

These routines were accessed through the *scipy.linalg.lstsq* wrapper function from the Scipy library.

For these tests, we consider the computation from scratch of the least-squares solution X_n verifying:

$$R_{\mathcal{N}}(n, m, r)X_n + \epsilon_n = R_{\mathcal{N}}(n, 1, 1) \tag{50}$$

where $R_{\mathcal{N}}(n, m, r) \in \mathbb{R}^{n \times m}$ is a pseudo-random matrix with rank r and whose elements are standardized ($\mu = 0$, $\sigma = 1$)[7]. For reproducibility purposes, the pseudo-random number generator was reset before each $R_{\mathcal{N}}(n, m, r)$ computation.

In order to assess the scaling properties of these algorithms, we evaluate (using Python's *timeit* module) the time elapsed for solving equation 50 using various input sizes. In the following, it is assumed that sufficiently large matrices were used for each algorithm to approach its asymptotic behavior in terms of computational time dependency with respect to input size. In other words, we assume that the scaling properties observed reflect the computational complexity of the algorithms considered[8]. These analyses were performed using various ranges for the parameters n, m, and r, allowing for an empirical characterization of the computational complexities involved:

- Full-rank square matrices:

  $$r = n = m \tag{51}$$

  Figure 1 highlights a roughly n^3 scaling of the elapsed computational time for large enough matrices, for all algorithms.

  This corroborates the $O(n^3)$ asymptotic complexity of all algorithms in this case.

- Full-rank rectangular matrices:

  $$r = n \leq m \tag{52}$$

[7] While random matrices are generally non rank-deficient, rank-deficient pseudo-random matrices are obtained here from the dot product of adequate pseudo-random matrices generated using the normal $\mathcal{N}(0, 1)$ distribution

[8] Which is equivalent to assuming that, for each algorithm, the largest computational times are dominated by the highest order term for the variable being considered.

When fixing the number of rows n, Figure 2 highlights a roughly linear scaling (with respect to m) of the elapsed computational time for large enough matrices, for all algorithms[9].

Since square matrices are a special case of rectangular matrices, this result is expected to be compatible with the scaling properties found for square matrices, if we assume a similar processing of both inputs. Therefore, in the case of full-rank rectangular matrices (with $n \leq m$), only an empirical mn^2 scaling (with respect to n and m) can be deduced from these experiments.

This supports a $O(mn^2)$ asymptotic complexity of all algorithms in this case.

- Rank-deficient square matrices:

$$r \leq n = m \tag{53}$$

At fixed rank $r = 100$, with square matrices of increasing size, Figure 3 nicely highlights the particular scaling (with respect to n) of rank-Greville, compared to the other commonly used algorithms. Indeed, all LAPACK-based algorithms display a roughly n^3 empirical scaling (comparable with results found for full-rank square matrices), while rank-Greville was found to display a roughly n^2 scaling of the elapsed computational time for large enough matrices.

Using a similar argument as before, and assuming rank-deficient matrices are not treated differently, these results are expected to be compatible with the previously found scaling properties. Therefore, the only empirical scaling (with respect to r, n and m) compatible with all previous experiments is in mnr for rank-Greville and mn^2 for all other algorithms.

These empirical findings demonstrate the distinctive ability of rank-Greville to take advantage of rank-deficiency, in order to reach an $O(mnr)$ asymptotic complexity.

- Optimal (time-efficiency wise) applicability domains:

$$r \leq n \leq m \tag{54}$$

Figure 4 illustrates which algorithm is the fastest (and by which margin) for a range of parameters ratios. Even though not specifically designed for solving the linear least-squares problem from scratch, the rank-Greville algorithm appears to be more efficient than other LAPACK solvers, but only when the observations matrix has particularly low rank $r \lesssim 0.15 \times \min(n, m)$.

[9] Additional tests at $m > 4 \times 10^5$ seem to confirm the asymptotic linear dependency on m for the DGELSS solver.

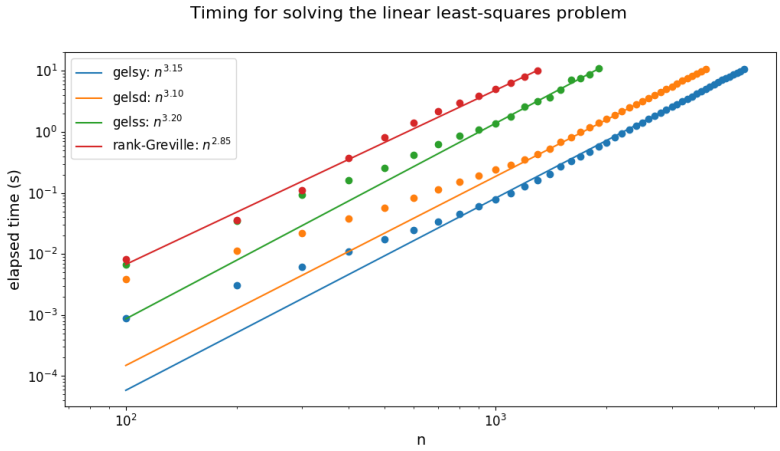

Figure 1: Timings for solving the linear least-squares problem on a random full-rank square observations matrix $R_{\mathcal{N}}(n, n, n)$. The asymptotic dependency with respect to n is fitted on the last points and reported in the legend.

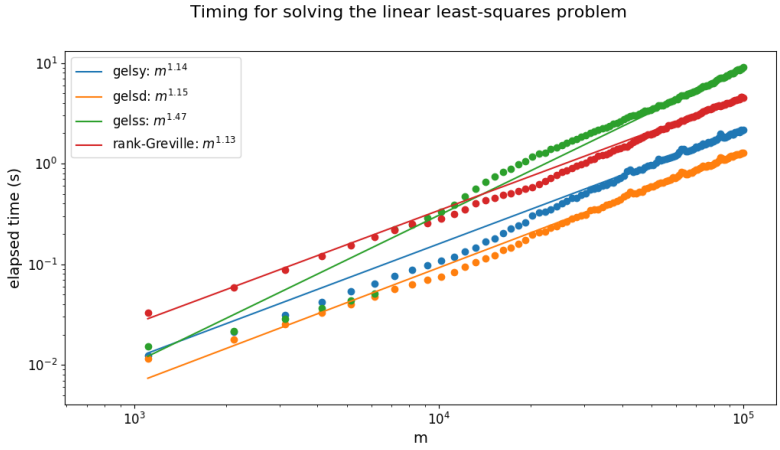

Figure 2: Timings for solving the linear least-squares problem on a random full-rank observations matrix $R_{\mathcal{N}}(n, m, n)$ with a fixed number of observations $n = 100$. The asymptotic dependency with respect to m is fitted on the last points and reported in the legend.

These tests confirm the lowest $O(mnr)$ asymptotic complexity of the rank-Greville algorithm compared with other LAPACK solvers ($O(m^2n)$ or $O(mn^2)$) for solving the least-squares problem from scratch. We note, nonetheless, that rank-Greville has a larger pre-factor, in part due to the additional work of maintaining a constantly up-to-date minimum-norm least-squares solution, typical of RLS solvers.

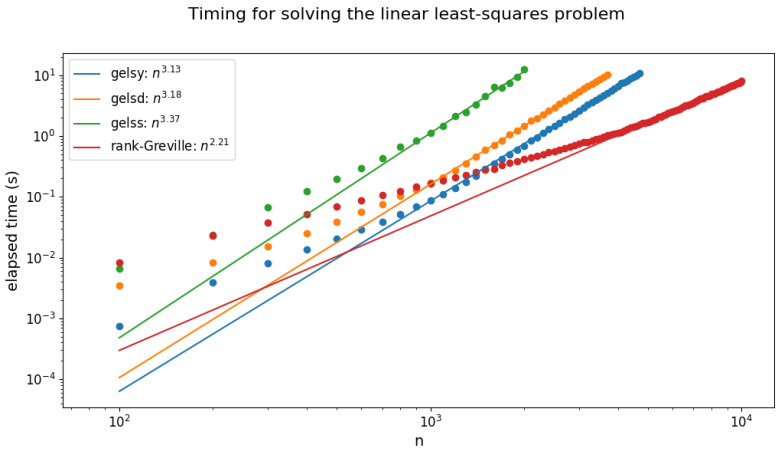

Figure 3: Timings for solving the linear least-squares problem on a random rank-deficient square observations matrix $R_{\mathcal{N}}(n, n, r)$ with a fixed rank $r = 100$. The asymptotic dependency with respect to n is fitted on the last points and reported in the legend.

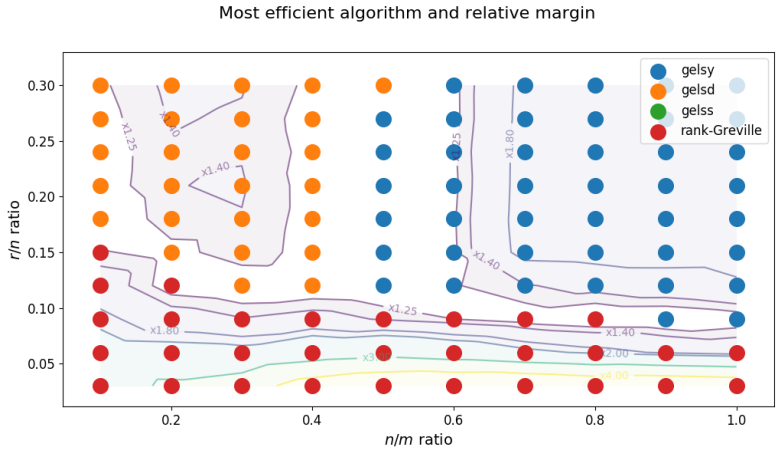

Figure 4: The fastest algorithm is represented for various n/m and r/n ratios, with $m = 4000$. The contour plot represents the interpolated relative margin by which an algorithm is the fastest (e.g. $\times 1.25$ means that the execution time for the second fastest algorithm was 1.25 times larger than for the fastest one).

4.2. Numerical stability

In this section we evaluate the numerical stability achieved by the approach described in this paper for computing the pseudoinverse.

For a more meaningful analysis, we compare our rank-Greville algorithm with standard algorithms from the LAPACK library described in 4.1, and other related algorithms:

- "Cholesky" refers to the *scipy.linalg.cho_solve* solver applied to the Cholesky factorization of $M = A^\top A$.
- "Greville" refers to a basic implementation of the original Greville algorithm[9, 21].
- "orthogonal" refers to the orthogonal variant described in section 2.2 with rescaling factors $\alpha_{n+1} = 1$.
- "orthonormal" refers to the orthonormal variant described in section 2.2 (i.e. orthogonal variant with rescaling factors $\alpha_{n+1} = ||\Gamma_{n+1}^{rej}||_2^{-1}$).

In order to assess the numerical stability of the algorithms described in this paper, we rely on the measures defined in [22, 20]:

- The stability factor of an algorithm with respect to the computation of the pseudoinverse A^+ of A is given by:

$$e_{\text{algo}} = \frac{\left|\left|A^+_{\text{algo}} - A^+\right|\right|_2}{\epsilon_M \, ||A^+||_2 \, \kappa_2(A)} \tag{55}$$

  where A^+_{algo} is the pseudoinverse of A computed by the algorithm, A^+ is the exact pseudoinverse, $||\cdot||_2$ is the 2-norm (e.g. for a matrix A, $||A||_2 = \max(\sigma(A))$ is the largest singular value of A), $\kappa_2(A) = \frac{\max(\sigma(A))}{\min(\sigma(A))}$ is the condition number of A and ϵ_M is the machine precision.

  This estimator is related to the forward stability of such algorithm, and should be bounded by a small constant depending on m and n.

- Similarly, we refer to the residual error as:

$$res_{\text{algo}} = \frac{\left|\left|A^+_{\text{algo}} A - I\right|\right|_2}{||A||_2 \left|\left|A^+_{\text{algo}}\right|\right|_2} \tag{56}$$

  where I is the identity.

  This estimator is related to the mixed forward-backward stability of such algorithm, and should be of the order of the machine precision ϵ_M.

During the test, the machine precision used corresponds to $\epsilon_M \approx 2.22 \times 10^{-16}$.

This evaluation was performed empirically, using the matrices defined in [22]:

- Pascal matrices $P(n) \in \mathbb{Z}^{n \times n}$.

  These are full-rank square matrices whose inverses $P(n)^{-1} \in \mathbb{Z}^{n \times n}$ can be computed exactly since their elements are also integers.

  Empirical results are reported in Tables 1 and 2. We found Greville-like algorithms, as well as the Cholesky decomposition to be orders of magnitudes less robust than the standard LAPACK solvers, with respect to both numerical stability indicators. Nonetheless, one should note that, as expected, when using *fractions.Fraction* based numerical representation, the rank Greville algorithm and its orthogonal variant were able to compute the exact inverse.

- Random matrices $R_{\mathcal{N}} \in \mathbb{R}^{3n \times n}$ whose elements are sampled from the normal $\mathcal{N}(0, 1)$ distribution.

  Pseudoinverses generated by the DGELSY solver were used as reference for computing the stability factor since they display the lowest residual error and the empirical results are reported in Tables 3 and 4. We also found that the numerical stability indicators for the Greville-like algorithms are significantly dependent on the seed used by the pseudorandom number generator, unlike other algorithms tested.

- Random ill-conditioned matrices ${R_{\mathcal{N}}}^4 \in \mathbb{R}^{n \times n}$, taken as the fourth power of random square matrices $R_{\mathcal{N}} \in \mathbb{R}^{n \times n}$ whose elements are sampled from the $\mathcal{N}(0, 1)$ distribution.

  Similarly as above, we used for pseudoinverse reference $\left({R_{\mathcal{N}}}^4\right)^+ = \left({R_{\mathcal{N}}}^+\right)^4$, the fourth power of the pseudoinverse generated by the DGELSY solver.

  Empirical results are reported in Tables 5 and 6, which show that for these ill-conditioned matrices, the Cholesky-based solver is, overall, the least stable algorithm tested herein.

- Random matrices $USV^\top$, where $U \in \mathbb{R}^{5n \times n}$ and $V \in \mathbb{R}^{n \times n}$ are random column orthogonal matrices and $S = diag(1, 2^{\frac{1}{2}}, \ldots, 2^{\frac{n-1}{2}})$.

  In this case, $(USV^\top)^+ = VS^{-1}U^\top$ was used for pseudoinverse reference.

  Empirical results are reported in Tables 7 and 8. For these tests, the *rcond*/*eps* parameter was set to 10^{-8}. This was required by Greville-like algorithms to reach a reasonable solution.

- Kahan matrices[23] $K(c, s) \in \mathbb{R}^{n \times n}$ with $c^2 + s^2 = 1$ and $n = 100$.

  An explicit formula is available for the inverse[23], and was used as pseudoinverse reference.

  Empirical results are reported in Table 9. Unlike what was reported in [22], we did not find the pure SVD-based solver to perform significantly worse than other LAPACK solvers. Furthermore, after setting the *rcond*/*eps* parameter low enough to tackle the extreme ill-conditionality of the Kahan matrices (i.e. $rcond < \kappa_2(A)^{-1}$), all algorithms (except Cholesky)

Table 1: Empiric stability factors associated with the pseudoinverse computation of Pascal matrices $P(n)$.

n	$\kappa_2(A)$	$e_{\text{orthonormal}}$	$e_{\text{orthogonal}}$	$e_{\text{rank-Greville}}$	e_{Greville}	e_{Cholesky}	e_{gelsy}	e_{gelsd}	e_{gelss}
4	6.92e+2	8.80e−2	1.11e−1	1.67e+0	4.73e−2	5.88e+0	3.86e−2	6.67e−2	6.76e−2
6	1.11e+5	7.18e+2	1.06e+2	2.12e+2	6.58e+2	2.44e+3	5.04e−3	4.74e−2	4.74e−2
8	2.06e+7	3.42e+5	1.93e+3	2.18e+4	2.08e+5	1.59e+5	3.86e−3	8.55e−3	8.55e−3
10	4.16e+9	1.08e+6	1.08e+6	1.08e+6	1.08e+6	7.34e+5	3.00e−4	1.86e−3	1.86e−3

Table 2: Empiric residual errors associated with the pseudoinverse computation of Pascal matrices $P(n)$.

n	$\kappa_2(A)$	$res_{\text{orthonormal}}$	$res_{\text{orthogonal}}$	$res_{\text{rank-Greville}}$	res_{Greville}	res_{Cholesky}	res_{gelsy}	res_{gelsd}	res_{gelss}
4	6.92e+2	1.83e−15	5.99e−16	3.85e−16	1.66e−17	4.01e−15	5.29e−17	5.39e−17	5.42e−17
6	1.11e+5	1.81e−13	4.16e−14	4.71e−14	1.46e−13	1.04e−12	2.49e−17	4.77e−17	4.47e−17
8	2.06e+7	7.60e−11	6.17e−13	4.84e−12	4.61e−11	9.24e−11	6.06e−17	3.16e−17	1.58e−17
10	4.16e+9	1.42e−9	1.61e−9	1.37e−9	1.54e−9	2.35e−8	4.78e−17	3.26e−17	1.54e−17

performed relatively well, with the QR-based LAPACK solver performing the best.

The algorithms described in this paper (rank-Greville and variants), including the original Greville algorithm, perform roughly equivalently in terms of numerical stability. Furthermore, the stability of these Greville-like algorithms seems much less dependent on $\kappa_2(A)$ than the Cholesky based algorithm. As expected, we found these algorithms to be neither mixed forward-backward nor forward stable. As a consequence, the much more robust LAPACK least-squares solvers are to be recommended when numerical stability is crucial. However, the stability of Greville-like algorithms are competitive compared to the Cholesky-based LS solvers.

A compromise between update efficiency and numerical stability can be searched among the full QR decomposition updating algorithms[1], or even faster, stable updating algorithms for Gram-Schmidt QR factorization[24, 1].

Table 3: Empiric stability factors associated with the pseudoinverse computation of random matrices $R_{\mathcal{N}} \in \mathbb{R}^{3n\times n}$ with elements distributed from $\mathcal{N}(0,1)$.

n	$\kappa_2(A)$	$e_{\text{orthonormal}}$	$e_{\text{orthogonal}}$	$e_{\text{rank-Greville}}$	e_{Greville}	e_{Cholesky}	e_{gelsd}	e_{gelss}
4	2.44e+0	1.06e+2	7.46e+1	2.02e+1	2.84e+0	7.57e−1	2.08e+0	1.67e+0
6	2.25e+0	4.00e+0	4.98e+0	2.56e+0	1.05e+0	9.20e−1	1.35e+0	1.86e+0
8	2.74e+0	3.98e+0	4.43e+0	2.86e+0	2.23e+0	1.39e+0	1.81e+0	2.63e+0
10	2.84e+0	1.14e+3	4.60e+2	3.18e+2	2.24e+1	1.54e+0	1.99e+0	2.21e+0

Table 4: Empiric residual errors associated with the pseudoinverse computation of random matrices $R_{\mathcal{N}} \in \mathbb{R}^{3n\times n}$ with elements distributed from $\mathcal{N}(0,1)$.

n	$\kappa_2(A)$	$res_{\text{orthonormal}}$	$res_{\text{orthogonal}}$	$res_{\text{rank-Greville}}$	res_{Greville}	res_{Cholesky}	res_{gelsy}	res_{gelsd}	res_{gelss}
4	2.44e+0	3.45e−14	2.55e−14	6.36e−15	7.20e−16	1.25e−16	2.24e−16	4.38e−16	4.68e−16
6	2.25e+0	8.53e−16	1.69e−15	7.61e−16	2.09e−16	2.32e−16	2.24e−16	4.10e−16	6.77e−16
8	2.74e+0	1.24e−15	1.05e−15	3.88e−16	3.39e−16	2.77e−16	2.48e−16	4.96e−16	6.16e−16
10	2.84e+0	4.75e−13	1.73e−13	3.73e−14	3.22e−15	4.36e−16	2.64e−16	6.33e−16	8.02e−16

Table 5: Empiric stability factors associated with the pseudoinverse computation of random ill-conditioned matrices $R^4_{\mathcal{N}} \in \mathbb{R}^{n\times n}$.

n	$\kappa_2(A)$	$e_{\text{orthonormal}}$	$e_{\text{orthogonal}}$	$e_{\text{rank-Greville}}$	e_{Greville}	e_{Cholesky}	e_{gelsy}	e_{gelsd}	e_{gelss}
6	3.39e+2	5.52e+0	3.04e+0	1.39e+1	8.87e+0	3.72e+1	5.59e−2	6.86e−2	7.36e−2
8	1.08e+2	1.50e+0	3.07e+0	4.16e+0	6.85e+0	2.40e+0	1.66e−1	1.58e−1	1.72e−1
10	2.92e+7	3.53e+0	2.62e+0	8.25e+0	5.62e+0	6.60e+5	2.07e−1	1.49e−2	1.49e−2
12	5.78e+4	2.47e+1	1.64e+1	2.41e+2	3.06e+1	1.62e+3	2.80e−2	7.38e−2	7.38e−2

Table 6: Empiric residual errors associated with the pseudoinverse computation of random ill-conditioned matrices $R^4_{\mathcal{N}} \in \mathbb{R}^{n\times n}$.

n	$\kappa_2(A)$	$res_{\text{orthonormal}}$	$res_{\text{orthogonal}}$	$res_{\text{rank-Greville}}$	res_{Greville}	res_{Cholesky}	res_{gelsy}	res_{gelsd}	res_{gelss}
6	3.39e+2	1.40e−15	7.32e−16	3.38e−15	2.03e−15	1.01e−14	5.06e−17	8.23e−17	4.55e−17
8	1.08e+2	8.97e−16	1.55e−15	1.05e−15	1.63e−15	2.90e−15	1.02e−16	3.00e−16	2.79e−16
10	2.92e+7	1.57e−15	8.01e−16	1.83e−15	1.25e−15	9.58e−10	6.37e−17	1.71e−16	1.58e−16
12	5.78e+4	6.09e−15	5.01e−15	5.36e−14	6.90e−15	1.55e−12	4.71e−17	9.97e−17	1.36e−16

Table 7: Empiric stability factors associated with the pseudoinverse computation of random matrices $USV^\top \in \mathbb{R}^{5n\times n}$, where U and V are random column orthogonal matrices and $S = diag(1, 2^{\frac{1}{2}}, \ldots, 2^{\frac{n-1}{2}})$.

n	$\kappa_2(A)$	$e_{\text{orthonormal}}$	$e_{\text{orthogonal}}$	$e_{\text{rank-Greville}}$	e_{Greville}	e_{Cholesky}	e_{gelsy}	e_{gelsd}	e_{gelss}
10	2.26e+1	6.72e+0	4.60e+0	4.19e+1	1.09e+1	1.85e+0	2.77e−1	2.94e−1	2.89e−1
15	1.28e+2	1.63e+2	1.60e+2	2.40e+3	2.99e+3	1.16e+1	1.70e−1	2.85e−1	2.93e−1
20	7.24e+2	1.57e+2	2.41e+2	1.52e+4	1.07e+4	4.24e+1	1.60e−1	1.85e−1	1.85e−1

Table 8: Empiric residual errors associated with the pseudoinverse computation of random matrices $USV^\top \in \mathbb{R}^{5n\times n}$, where U and V are random column orthogonal matrices and $S = diag(1, 2^{\frac{1}{2}}, \ldots, 2^{\frac{n-1}{2}})$.

n	$\kappa_2(A)$	$res_{\text{orthonormal}}$	$res_{\text{orthogonal}}$	$res_{\text{rank-Greville}}$	res_{Greville}	res_{Cholesky}	res_{gelsy}	res_{gelsd}	res_{gelss}
10	2.26e+1	2.68e−15	4.25e−15	2.52e−15	5.92e−16	1.70e−15	1.01e−16	1.62e−16	1.26e−16
15	1.28e+2	1.67e−14	9.69e−15	9.20e−14	2.59e−13	7.37e−15	7.58e−17	1.19e−16	1.30e−16
20	7.24e+2	1.65e−14	6.11e−14	6.04e−13	4.93e−13	3.53e−14	4.31e−17	9.96e−17	5.61e−17

Table 9: Empiric residual errors associated with the pseudoinverse computation of Kahan matrices $K(c, s) \in \mathbb{R}^{100\times100}$, with $c^2 + s^2 = 1$.

c	$\kappa_2(A)$	$res_{\text{orthonormal}}$	$res_{\text{orthogonal}}$	$res_{\text{rank-Greville}}$	res_{Greville}	res_{Cholesky}	res_{gelsy}	res_{gelsd}	res_{gelss}
0.10	5.42e+4	5.57e−17	3.00e−17	3.50e−17	2.64e−17	3.40e−13	5.24e−17	7.61e−17	1.33e−16
0.15	1.13e+7	1.11e−17	2.16e−17	1.03e−17	1.19e−17	1.60e−10	1.52e−17	5.61e−17	7.97e−17
0.20	2.18e+9	8.10e−18	7.96e−18	2.42e−18	2.41e−18	failure	7.40e−18	6.30e−17	5.69e−17
0.25	4.37e+11	2.07e−18	1.06e−18	1.14e−18	1.17e−18	1.61e−6	8.74e−18	4.85e−17	4.39e−17
0.30	9.77e+13	3.01e−19	4.31e−19	1.92e−19	1.94e−19	7.98e−4	1.82e−19	1.33e−16	2.43e−17
0.35	2.57e+16	3.57e−20	4.27e−20	2.73e−20	2.18e−20	failure	5.05e−20	9.43e−18	2.03e−18
0.40	8.36e+18	3.46e−21	3.51e−21	3.09e−21	2.55e−21	1.02e−4	1.08e−20	3.17e−19	3.04e−19

5. Conclusion

In this paper, we first derive a simple explicit formula for the recursive least-squares problem using a general rank decomposition update scheme. Based on this, we devise a transparent, Greville-like algorithm. We also introduce two variants bridging the gap between Greville's algorithm and QR-based least-squares solvers. In contrast to Greville's algorithm, we maintain a rank decomposition at each update step. This allows us to exploit rank-deficiency to reach an asymptotic computational complexity of $O(mr)$ for updating a least-squares solution when adding an observation, leading to a total complexity of $O(mnr)$ for computing the full least-squares solution. This complexity is lower than Greville's algorithm or any commonly available solver tested, even though a truncated QR factorization based solver can achieve such a $O(mnr)$ bound for computing the full least-squares solution[25]. Nonetheless, a $O(mr)$ bound for the least-squares solution update step is, to our knowledge, lower than those achieved by the more sophisticated updating algorithms explicitly reported in the literature. We have implemented these algorithms in Python3, using Numpy. This publicly available implementation offers a recursive least-squares solver ($O(mr)$ operations per update), with optional pseudoinverse ($O(mn)$) and covariance support ($O(m^2)$). The numerical stability of these Greville-like algorithms were empirically found to be significantly inferior compared to common LAPACK solvers. However, it is important to note that the algebraic simplicity of some of these Greville-like methods make them compatible with exact numerical representation, without the need to use symbolic computing.

References

[1] Bjorck A. Numerical Methods for Least Squares Problems. Other Titles in Applied Mathematics; Society for Industrial and Applied Mathematics; 1996. ISBN 9780898713602. URL: https://doi.org/10.1137/1.9781611971484.

[2] Penrose R. On best approximate solutions of linear matrix equations. Mathematical Proceedings of the Cambridge Philosophical Society 1956;52(1):17–19. doi:10.1017/S0305004100030929.

[3] Rakha MA. On the Moore–Penrose generalized inverse matrix. Applied Mathematics and Computation 2004;158(1):185–200. URL: https://linkinghub.elsevier.com/retrieve/pii/S0096300303009998. doi:10.1016/j.amc.2003.09.004.

[4] Toutounian F, Ataei A. A new method for computing Moore–Penrose inverse matrices. Journal of Computational and Applied Mathematics 2009;228(1):412–7. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377042708005062. doi:10.1016/j.cam.2008.10.008.

[5] Moore E. On the reciprocal of the general algebraic matrix. Bull Amer Math Soc 1920;26(9):394–5. URL: https://doi.org/10.1090/S0002-9904-1920-03322-7. doi:10.1090/S0002-9904-1920-03322-7.

[6] Penrose R. A generalized inverse for matrices. Mathematical Proceedings of the Cambridge Philosophical Society 1955;51(3):406–413. doi:10.1017/S0305004100030401.

[7] Björck Å. Least Squares Problems. Boston, MA: Springer US. ISBN 978-0-387-74759-0; 2009, p. 1856–66. URL: https://doi.org/10.1007/978-0-387-74759-0_329. doi:10.1007/978-0-387-74759-0_329.

[8] Courrieu P. Fast computation of moore-penrose inverse matrices. ArXiv 2005;abs/0804.4809.

[9] Greville TNE. Some applications of the pseudoinverse of a matrix. SIAM Review 1960;2(1):15–22. URL: https://doi.org/10.1137/1002004. doi:10.1137/1002004. arXiv:https://doi.org/10.1137/1002004.

[10] Albert A, Sittler RW. A method for computing least squares estimators that keep up with the data. Journal of the Society for Industrial and Applied Mathematics Series A Control 1965;3(3):384–417. URL: https://doi.org/10.1137/0303026. doi:10.1137/0303026. arXiv:https://doi.org/10.1137/0303026.

[11] Le Gall F. Faster algorithms for rectangular matrix multiplication. In: 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science. 2012, p. 514–23. doi:10.1109/FOCS.2012.80.

[12] Mirsky L. An Introduction to Linear Algebra. Dover Books on Mathematics; Dover; 1990. ISBN 9780486664347. URL: https://books.google.fr/books?id=ULMmheb26ZcC.

[13] Greene W. Econometric Analysis. Prentice Hall; 2003. ISBN 9780130661890.

[14] Van Rossum G, Drake FL. Python 3 Reference Manual. Paramount, CA: CreateSpace; 2009. ISBN 1441412697, 9781441412690.

[15] Pérez F, Granger BE. IPython: a system for interactive scientific computing. Computing in Science and Engineering 2007;9(3):21–9. URL: https://ipython.org. doi:10.1109/MCSE.2007.53.

[16] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 2020;17:261–72. doi:https://doi.org/10.1038/s41592-019-0686-2.

[17] Oliphant TE. Guide to NumPy. 2nd ed.; USA: CreateSpace Independent Publishing Platform; 2015. ISBN 151730007X, 9781517300074.

[18] Walt Svd, Colbert SC, Varoquaux G. The numpy array: A structure for efficient numerical computation. Computing in Science & Engineering 2011;13(2):22–30. URL: https://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37. doi:10.1109/MCSE.2011.37.

[19] Anderson E, Bai Z, Bischof C, Blackford LS, Demmel J, Dongarra J, et al. LAPACK Users' Guide. Third ed.; Society for Industrial and Applied Mathematics; 1999. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898719604. doi:10.1137/1.9780898719604. arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780898719604.

[20] Higham NJ. Accuracy and Stability of Numerical Algorithms. Second ed.; Society for Industrial and Applied Mathematics; 2002. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898718027. doi:10.1137/1.9780898718027. arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780898718027.

[21] Ben-Israel A, Greville TN. Generalized inverses: theory and applications; vol. 15. Springer Science & Business Media; 2003.

[22] Smoktunowicz A, Wrobel I. Numerical aspects of computing the moore-penrose inverse of full column rank matrices. BIT Numerical Mathematics 2012;52(2):503–24.

[23] Kahan W. Numerical linear algebra. Canadian Mathematical Bulletin 1966;9(5):757–801. doi:10.4153/CMB-1966-083-2.

[24] Daniel JW, Gragg WB, Kaufman L, Stewart GW. Reorthogonalization and stable algorithms for updating the gram-schmidt qr factorization. Mathematics of Computation 1976;30(136):772–95.

[25] Businger P, Golub GH. Linear least squares solutions by householder transformations. Numer Math 1965;7(3):269–276. URL: https://doi.org/10.1007/BF01436084. doi:10.1007/BF01436084.