Randomized Algorithms for Computing the Rank Profile of a Matrix over a Finite Field

by

Alex Krueger

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Arne Storjohann

Waterloo, Ontario, Canada, 2014

Abstract

Randomized algorithms are given for the problem of computing the row rank profile of a matrix $A \in \mathsf{K}^{n \times m}$, for K, a field. We give an algorithm that computes the row rank profile of A in $mr^2 + \frac{8}{3}r^3 + O(mr + r^2 + |A|)$ field operations from K, where $r = \text{rank}(A)$ and $|A|$ denotes the number of nonzero elements of A. The algorithm is Monte Carlo randomized and returns the correct solution with probability $(1 - \frac{1}{\#\mathsf{K}})^r$. Improvements to the algorithm are discussed to guarantee that the algorithm produces the correct profile with probability at least 1/2.

We give a Monte Carlo certification for the rank profile that takes as input the matrix A, a claimed row rank profile $\mathcal{P}$, and a certificate $(\mathcal{Q}, ((R_r, L_r), (R_{r-1}, L_{r-1}), \cdots, (R_1, L_1)))$, where $\mathcal{Q}$ is a list of column indices, and $((R_r, L_r), \cdots, (R_1, L_1))$ is claimed to be a unique decomposition of the claimed inverse of the $r \times r$ submatrix of A composed of the rows indicated in $\mathcal{P}$ and the columns indicated in $\mathcal{Q}$. The certification runs in $4r^2 + O(r + |A|)$ field operations from K. If $\mathcal{P}$ is in fact the row rank profile of A and B is the claimed inverse, then the algorithm will return TRUE. Otherwise, the certification will incorrectly return TRUE with probability at most $2/\#\mathsf{K}$. Improvements are discussed to ensure that that $\mathcal{P}$ is incorrectly certified as the row rank profile of A with probability at most 1/2.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Arne Storjohann, for all the guidance, support and encouragement. I would also like to thank Prof. Romain Lebreton for his feedback on this project.

Dedication

For my family.

Table of Contents

Chapter 1

Introduction

Let $A \in \mathsf{K}^{n\times m}$ for some finite field K with rank r. The row rank profile of a matrix A is the lexicographically minimal list $\mathcal{P} = [i_1, \ldots, i_r]$ such that the rows of $A^{\mathcal{P}}$, the $r \times m$ submatrix of A consisting of the rows listed in $\mathcal{P}$, are linearly independent.

The row rank profile is the list of rows containing the pivot elements when the matrix is transformed to column echelon form using the following elementary column operations: interchanging two columns, adding a multiple of one column to another column. For example, consider a 10×10, rank 6 matrix. Let $\bullet$ denote a known nonzero element and $*$ denote a possible nonzero element, and blank sections denote blocks of zeroes. Suppose the matrix has the following column echelon form

$$\begin{bmatrix}
\bullet &&&&&&&&& \\
* & \bullet &&&&&&&& \\
* & * &&&&&&&& \\
* & * & \bullet &&&&&&& \\
* & * & * &&&&&&& \\
* & * & * & \bullet &&&&&& \\
* & * & * & * & \bullet &&&&& \\
* & * & * & * & * &&&&& \\
* & * & * & * & * & \bullet &&&& \\
* & * & * & * & * & * &&&&
\end{bmatrix}.$$

The row rank profile is also the list of indices of the nonzero rows of A after transforming the matrix using the following elementary row operations: adding a multiple of one row to lower row. For example, suppose the matrix has the following form if Gaussian elimination (row operations) without pivoting is performed

$$\begin{bmatrix}
\bullet & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast \\
 & \bullet & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast \\
 & & & & & & & & & \\
 & & & & & & & & & \bullet \\
 & & & & & & & & & \\
 & & & & & \bullet & \ast & \ast & \ast & \\
 & & \bullet & \ast & \ast & & \ast & \ast & \ast & \\
 & & & & & & & & & \\
 & & & \bullet & \ast & & \ast & \ast & \ast & \\
 & & & & & & & & &
\end{bmatrix}.$$

In either case, the row rank profile $\mathcal{P} = [1, 2, 4, 6, 7, 9]$ is revealed.

To compute $r = \text{rank}(A)$, Gaussian elimination can be run to reduce either the rows or columns in A in $O(nmr)$ field operations from K. The algorithms presented herein use the idea of doing Gaussian elimination on the rows without pivoting to find the first r linearly independent rows.

Note that if a vector a is sparse, then for any vector x, the dot product $a^\top x$ can be computed in $O(|a|)$ operations by only considering the nonzero terms in a. Thus, the matrix-vector product Ax can be computed in $O(|A|)$ field operations from K, which may be considerably faster than the typical bound of $O(nm)$ field operations. We will therefore exploit the sparsity of A to reduce the runtime of the algorithms presented herein.

In Section 1.1, we will first present previous algorithms. We will then introduce our new results and then present some notation used throughout this paper.

1.1 Monte Carlo Computation

Some work has been done to develop algorithms to solve the problem of computing the row rank profile or a matrix $A \in \mathsf{K}^{n\times m}$ with $\text{rank}(A) = r$, where r may not be known and is computed as the length of the row rank profile. Note that since computing the row rank profile of A involves computing linearly independent rows, the work required is similar to the work required to solve the linear system $Ax = b$, for some vector $b \in \mathsf{K}^{n\times 1}$. This is the approach Storjohann and Yang take in [4]. Counting the number of field operations

$(+, -, \times, \div)$ from K required to run the algorithm, classical Gaussian elimination can be used to solve the system with a runtime of $O(nmr)$ field operations. If the pivot rows are carefully chosen to ensure that the row indices form a lexicographically minimal list of maximal size, then the resulting Gaussian elimination process can be used to generate the row rank profile of A. Note that since $r \leq \min(m, n)$, many of the algorithms focus on having the runtime depend more on r, rather than m and n, the dimension of A. This is particularly useful for low rank applications, where $r \ll \min(m, n)$, since in these cases $r^3 \ll mnr$. Thus an algorithm whose runtime depends also on the rank will be faster in certain instances than an algorithm that depends only on the dimension.

Storjohann and Yang [4] introduce the use of a linear independence oracle. Given a matrix $R \in \mathsf{K}^{n\times m}$, a linear independence oracle is a binary tree T describing various linear combinations of columns of A. Given a vector $v \in \mathsf{K}^{1\times n}$, testing whether vR is the zero vector can be done by computing dot products with the vectors stored at the nodes in T. If $vR \neq 0$, the search from the root to the bottom level returns the index of the first nonzero element of vR. Using linear independence oracles, a Las Vegas algorithm `RandomOracleSolve` [4] is presented that takes as input a rank r matrix $A \in \mathsf{K}^{n\times m}$, where r is unknown, as well as a vector $b \in \mathsf{K}^{n\times 1}$, and computes the solution $x \in \mathsf{K}^{m\times 1}$ such that $Ax = b$, or a certificate of inconsistency $u \in \mathsf{K}^{1\times n}$ such that $uA = 0$ and $ub \neq 0$. If A is dense, then `RandomOracleSolve` runs in $2r^3 + O(r^2(\log n + \log m) + r(n + m))$ field operations from K, where the logarithmic factors are from the construction of and search through T. Similarly, if A is sparse, then the runtime bound is $r^3 + O(r^2(\log n + \log m) + n + m + |A^{\mathcal{P}}| \log m + |A_{\mathcal{Q}}| \log n)$. As a side effect, this algorithm also computes a list of row indices $\mathcal{P}$ of length at most r such that the submatrix $A^{\mathcal{P}}$ is full rank, as well as a list of column indices $\mathcal{Q}$ such that the submatrix $A^{\mathcal{P}}_{\mathcal{Q}}$ has generic rank profile, where $A_{\mathcal{Q}}$ is the $n \times r$ submatrix of A made of the columns indicated in $\mathcal{Q}$. That is, the algorithm also finds a list $\mathcal{Q}$ such that every square leading submatrix of $A^{\mathcal{P}}_{\mathcal{Q}}$ is invertible. The Monte Carlo algorithm `ImprovedRankProfile` exploits this to generate the row rank profile of A. `ImprovedRankProfile` first generates a random vector $g \in \mathsf{K}^{m\times 1}$ and computes $b = Ag$. The matrix A and vector b are then passed into `RandomOracleSolve`, since there is a solution to the system $Ax = b$. With probability at least $(1 - \frac{1}{\#\mathsf{K}})^r (1 - \frac{r}{\#\mathsf{K}})^{\lceil \log_2 n\rceil + \lceil \log_2 m\rceil}$, the list $\mathcal{P}$ returned by `RandomOracleSolve` will be the row rank profile of A. If the field size is sufficiently small and the dimension or rank of A are sufficiently large, then the lower bound on the probability of correctness of the algorithm will be small, and even possibly less than 0. In the case of a small field, field extensions can then be used to guarantee a lower bound on the probability of correctness [4, Corollary 20].

1.2 Monte Carlo Certification

The idea of Monte Carlo certification is to allow verification of whether or not a claimed solution is correct in less time than it would take to compute the answer without the certificate. Certification takes as input the parameters for the problem, the claimed solution and a certificate, which is a tuple of useful results with some claimed properties.

For example, let $A \in \mathsf{K}^{n \times n}$ and $B \in \mathsf{K}^{n \times n}$. Suppose we are given $C \in \mathsf{K}^{n \times n}$ with the claim that $C = AB$. One could simply compute AB in $O(n^3)$ field operations and compare the results. However, directly computing the product is costly and does not make use of the fact that C may be the product of A and B. Freivalds' Algorithm [2] exploits the fact that if $AB = C$, then for every $c \in \mathbb{Z}_2^{n \times 1}$, $ABc = Cc$, which can be computed in $O(n^2)$ field operations. However, if $AB \neq C$, the probability that $ABc = Cc$ is at most $1/2$. Moreover, if using p certificates c, the probability that $ABc = Cc$ for all p choices c is less than $1/2^p$, assuming $AB \neq C$. No certificate tuple is required for this verification.

Similarly, Kaltofen, Nehring and Saunders [1] describe certification for the nonsingularity of a matrix $A \in \mathbb{Z}^{n \times n}$ using a certificate (p, B) with the claim that p is prime and $B = A^{-1} \bmod p$. Note that if A is singular, then no such B can exist for any prime p. The matrix B can then be tested as the inverse of A mod p as in Freivalds' algorithm [1] by testing that $ABc = Ic$ for some $c \in \mathbb{Z}_p^{r \times 1}$. These tests can be run in $O(n^2)$ operations from $\mathbb{Z}_p$, which is linear in the size of the input. However, certification is Monte Carlo and may indicate that a solution is correct when it is not.

1.3 New Results

In Section 2.4, we give a new Monte Carlo algorithm, Algorithm 2 (`RandomRowRankProfile`), which computes the row rank profile of A. Algorithm 2 (`RandomRowRankProfile`) is a modification of the `ImprovedRankProfile` from [4]. Algorithm 2 (`RandomRowRankProfile`) eliminates the need for a linear independence oracle. As a result the algorithm runs in $mr^2 + \frac{8}{3}r^3 + O(mr + r^2 + |A|)$ field operations, whereas the algorithm in [4] has a runtime of $r^3 + O(r^2(\log n + \log m) + n + m + |A^{\mathcal{P}}| \log m + |A_{\mathcal{Q}}| \log n)$. Note that if $r = m$, the upper bound on the running time of `RandomRowRankProfile` is $\frac{11}{3}r^3 + O(r^2 + |A|)$. While the leading coefficient of the runtime is larger, and the leading term also depends on m, `RandomRowRankProfile` has the following advantages over `ImprovedRankProfile`.

- The run time has no logarithmic terms.

- The probability of correctness depends only on the rank of A and size of K, not on n or m.
- The probability of correctness of the algorithm is always positive over any field and for any rank. As a result, in case of a small field, field extensions are not necessary to increase the likelihood of correctness of the algorithm.
- There is no need to construct a linear independence oracle. The algorithm can be run in place.

In Section 3.2, we present a Monte Carlo certification algorithm, Algorithm 3`RandomRowProfileCertificate`, adapted from Algorithm 2`RandomRowRankProfile`. Given a claimed row rank profile $\mathcal{P}$, and a certificate $(\mathcal{Q}, B)$ with the claim that $B = (A_{\mathcal{Q}}^{\mathcal{P}})^{-1}$, the algorithm tests whether $\mathcal{P}$ is the row rank profile of A. The algorithm runs in $4r^2+O(r+|A|)$. If $\mathcal{P}$ is the row rank profile of A, then the algorithm returns TRUE with certainty. Otherwise, if $\mathcal{P}$ is not the row rank profile of A, then the algorithm will incorrectly confirm that it is the row rank profile of A with probability at most $2/\#\mathsf{K}$. Improvements to the algorithm are then discussed to guarantee that the probability that the algorithm falsely returns TRUE is lower than some desired upper bound. Thus, as with `RandomRowRankProfile`, field extensions are not necessary.

1.4 Notation

Recall that $\mathcal{P} = [i_1, \ldots, i_s]$ is a list of distinct positive integers and $\mathcal{Q} = [j_1, ..., j_s]$ be similarly defined. Let $\neg\mathcal{P} = [i \mid 1 \leq i \leq n,\ i \notin \mathcal{P}]$ and $\neg\mathcal{Q} = [j \mid 1 \leq j \leq m,\ j \notin \mathcal{Q}]$. Finally, when $s \geq 1$, let $\mathcal{P}_{-1} = [i_1, \ldots, i_{s-1}]$ and $\mathcal{Q}_{-1} = [j_1, \ldots, j_{s-1}]$, respectively.

Chapter 2

Improved Random Row Rank Profile

In this chapter we present a Monte Carlo randomized algorithm for computing the row profile of an input matrix $A \in \mathsf{K}^{n \times m}$ of unknown rank r. The algorithm runs in $mr^2 + \frac{8}{3}r^3 + O(mr + r^2 + |A|)$ time and produces the correct result with probability $(1 - \frac{1}{\#\mathsf{K}})^r$.

2.1 Useful Properties

The following properties are in [4]. Let $\mathcal{P} = [i_1, \ldots, i_s]$ and $\mathcal{Q} = [j_1, \ldots, j_s]$ be given. Suppose $A^{\mathcal{P}}_{\mathcal{Q}}$ and its leading $(s-1) \times (s-1)$ submatrix, $A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}}$, are invertible.

Lemma 1. *Let $i \in \neg\mathcal{P}$. Consider the augmented matrix*

$$\left[\begin{array}{c|c} A^{\mathcal{P}}_{\mathcal{Q}} & A^{\mathcal{P}}_{\neg\mathcal{Q}} \\ \hline A^{[i]}_{\mathcal{Q}} & A^{[i]}_{\neg\mathcal{Q}} \end{array}\right]. \tag{2.1}$$

Then $A^{[i]} \in$ Rowspace$(A^{\mathcal{P}})$ if and only if $A^{[i]}_{\neg\mathcal{Q}} - A^{[i]}_{\mathcal{Q}}(A^{\mathcal{P}}_{\mathcal{Q}})^{-1}A^{\mathcal{P}}_{\neg\mathcal{Q}} = 0$

Proof. A block Gaussian elimination of the matrix in (2.1) gives the following:

$$\left[\begin{array}{c|c} I & 0 \\ \hline -A^{[i]}_{\mathcal{Q}}(A^{\mathcal{P}}_{\mathcal{Q}})^{-1} & I \end{array}\right]\left[\begin{array}{c|c} A^{\mathcal{P}}_{\mathcal{Q}} & A^{\mathcal{P}}_{\neg\mathcal{Q}} \\ \hline A^{[i]}_{\mathcal{Q}} & A^{[i]}_{\neg\mathcal{Q}} \end{array}\right] = \left[\begin{array}{c|c} A^{\mathcal{P}}_{\mathcal{Q}} & A^{\mathcal{P}}_{\neg\mathcal{Q}} \\ \hline 0 & A^{[i]}_{\neg\mathcal{Q}} - A^{[i]}_{\mathcal{Q}}(A^{\mathcal{P}}_{\mathcal{Q}})^{-1}A^{\mathcal{P}}_{\neg\mathcal{Q}} \end{array}\right]. \tag{2.2}$$

Let $t := A^{[i]}_{\neg\mathcal{Q}} - A^{[i]}_{\mathcal{Q}}(A^{\mathcal{P}}_{\mathcal{Q}})^{-1}A^{\mathcal{P}}_{\neg\mathcal{Q}}$. Note that t is the row vector corresponding to $A^{[i]}_{\neg\mathcal{Q}}$ that results from eliminating $A^{[i]}_{\mathcal{Q}}$ using $A^{\mathcal{P}}$ as a block pivot. Thus, if $t = 0$ then $A^{[i]}$ is some linear combination of the rows of $A^{\mathcal{P}}$. If $t \neq 0$ then $A^{[i]}$ is linearly independent of the rows of $A^{\mathcal{P}}$. □

Let

$$A^{\mathcal{P}}_{\mathcal{Q}} = \left[\begin{array}{c|c} A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}} & u \\ \hline v & d \end{array}\right] \in \mathsf{K}^{s\times s}, \tag{2.3}$$

where $u := A^{\mathcal{P}_{-1}}_{[j_s]}$, $v := A^{[i_s]}_{\mathcal{Q}_{-1}}$, and $d := A^{[i_s]}_{[j_s]}$. The inverse of $A^{\mathcal{P}}_{\mathcal{Q}}$ is

$$\left[\begin{array}{c|c} B + (Bu)w(vB) & -(Bu)w \\ \hline -w(vB) & w \end{array}\right]. \tag{2.4}$$

where $B := (A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}})^{-1}$, and $w := (d - vBu)^{-1}$. The correctness of (2.4) as the inverse of (2.3) can be verified by direct computation.

Lemma 2. *Let $s \geq 1$. If $B = (A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}})^{-1}$ is precomputed, the inverse of $A^{\mathcal{P}}_{\mathcal{Q}}$ can be computed in $6s^2 + O(s)$ field operations from K.*

Proof. The products vB and Bu can be computed in $2s^2 + O(s)$ operations each, while the product $(Bu)w(vB)$ and sum $B+(Bu)w(Bv)$ can be computed in $s^2+O(s)$ field operations each. The rest of the operations are dot products, sums and matrix-scalar multiplications that can be done in $O(s)$ time each, giving a total runtime of $6s^2+O(s)$ field operations. □

Corollary 3. *Let $b^{\mathcal{P}} \in \mathsf{K}^{s\times 1}$. Assuming $Bb^{\mathcal{P}_{-1}}$, Bu and vB are precomputed, we can compute $(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}b^{\mathcal{P}}$ in $O(s)$ field operations from K by using the form given in (2.4), i.e.,*

$$(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}b^{\mathcal{P}} = \left[\begin{array}{c} Bb^{\mathcal{P}_{-1}} + (Bu)w((vB)b^{\mathcal{P}_{-1}}) - (Bu)wb^{[i_s]} \\ \hline wb^{[i_s]} - w(vB)b^{\mathcal{P}_{-1}} \end{array}\right]. \tag{2.5}$$

Proof. Given that Bu, vB, $Bb^{\mathcal{P}_{-1}}$ are precomputed, this update can be done in 2 dot products, 2 vector-scalar multiplications, 2 vector additions and subtractions, totalling $8s + O(1)$ operations. □

Corollary 4. *Consider the following product within (2.2):*

$$(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}A_{\neg\mathcal{Q}}^{\mathcal{P}}.$$

If $(A_{\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}})^{-1}A_{\neg\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}}$ is precomputed, then $(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}A_{\neg\mathcal{Q}}^{\mathcal{P}}$ can be computed by considering each column of $(A_{\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}})^{-1}A_{\neg\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}}$ separately, using (2.5), without needing to consider column j_s. This computation can be done in $8(m-s)s + O(m+s)$ field operations from K.

Proof. The proof follows from Corollary 3. Recall that $A_{\neg\mathcal{Q}}^{\mathcal{P}} \in \mathsf{K}^{(m-s)\times s}$, so each of the $m-s$ column vectors of size s can be updated in $8s + O(1)$ field operations, giving an overall cost of $8(m-s)s + O(m+s)$ field operations from K. □

2.2 A Deterministic Approach

The lemmas and corollaries from the previous section give rise to Algorithm 1 (`RowRank-Profile`) to compute the row rank profile of an input matrix $A \in \mathsf{K}^{n\times m}$.

The idea behind Algorithm 1 (`RowRankProfile`) is to use the criterion of Lemma 1 at each step to find the first row that is linearly independent from all previously found rows. The row index of the next linearly independent row is then added to $\mathcal{P}$ and the loop is iterated. The matrices B and $BA_{\neg\mathcal{Q}}^{\mathcal{P}}$ are used for each index s. Note that once a row $A^{[i]}$ is found to be dependent on some subset of the row rank profile $[i_1, \ldots, i_{s-1}]$, then $A^{[i]}$ will be dependent on $[i_1, \ldots, i_s]$. For this reason, once a row is examined, it is never reexamined.

Algorithm 1 RowRankProfile

```
Input: $A \in \mathsf{K}^{n\times m}$
Output: $\mathcal{P} = [i_1, \ldots, i_r]$, the row rank profile of $A$
  {$B$ stores $(A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$ after each step}
  {$C$ stores $(A^{\mathcal{P}}_{\mathcal{Q}})^{-1} A^{\mathcal{P}}_{\neg\mathcal{Q}}$}
  {Note that if $\mathcal{P} = [\ ]$ or $\mathcal{Q} = [\ ]$, the submatrices they denote have a dimension of 0}
  Initialize $\mathcal{P}, \mathcal{Q} \leftarrow [\ ]$,
  $B, C \leftarrow [\ ] \in \mathsf{K}^{0\times 0}$
  $s \leftarrow 1$
  for $i$ from 1 to $m$ do
    if $t := A^{[i]}_{\neg\mathcal{Q}} - A^{[i]}_{\mathcal{Q}} C \neq 0$ then
      Let $j \in \neg\mathcal{Q}$ such that $t_{[j]} \neq 0$ be minimal
      $v := A^{[i]}_{\mathcal{Q}}$, $u := A^{\mathcal{P}}_{[j]}$, $d := A^{[i]}_{[j]}$
      Precompute $vB$ and $w = (d - vBu)^{-1}$, extract $Bu := C_{[j]}$
      $i_s \leftarrow i$, $j_s \leftarrow j$
      $\mathcal{P} \leftarrow [\mathcal{P}, i_s]$, $\mathcal{Q} \leftarrow [\mathcal{Q}, j_s]$
      Update $B \leftarrow \left[\begin{array}{c|c} B + (Bu)w(vB) & -(Bu)w \\ \hline -w(vB) & w \end{array}\right]$
      Update $C$ similarly using Corollary (4)
      $s \leftarrow s + 1$
    end if
  end for
  return $\mathcal{P}$
```

The overall cost of updating B for every increment of s is,

$$\sum_{s=1}^{r}(6s^2 + O(s)) = 2r^3 + O(r^2)$$

The overall cost of updating $BA^{\mathcal{P}}_{\neg\mathcal{Q}}$ for every increment of s is,

$$\sum_{s=1}^{r}(8(m-s)s + O(m+s)) = 4mr^2 - \frac{8}{3}r^3 + O(mr + r^2)$$

Computing t at a given iteration i with associated step s is an $2ms - s^2 + O(m+s)$ operation. In the worst case, the row rank profile of A will be the first r rows, and then $s = r$ for $r+1 \le i \le n$. So computing t has an overall cost of

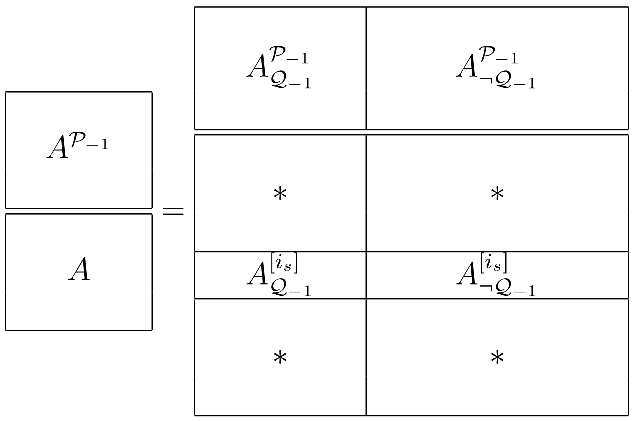
$$\sum_{s=1}^{r}(2ms - s^2 + O(m+s)) + \sum_{i=r+1}^{m}(2mr - r^2 + O(m+r)) = 2m^2r - 2mr^2 + \frac{1}{3}r^3$$

The vector Bu is precomputed as columns j of C, while the cost of computing vB at every step s is $2s^2 + O(s)$. The overall cost of computing vB is

$$\sum_{s=1}^{r}(2s^2 + O(s)) = \frac{2}{3}r^3 + O(r^2)$$

The amortized cost of multiplying $BA^{\mathcal{P}}_{\neg\mathcal{Q}}$ by each row of A is $O(m|A|)$. Algorithm 1 (`RowRankProfile`) therefore has cost $2m^2r + 2mr^2 + \frac{1}{3}r^3 + O(mr + r^2 + m|A|)$.

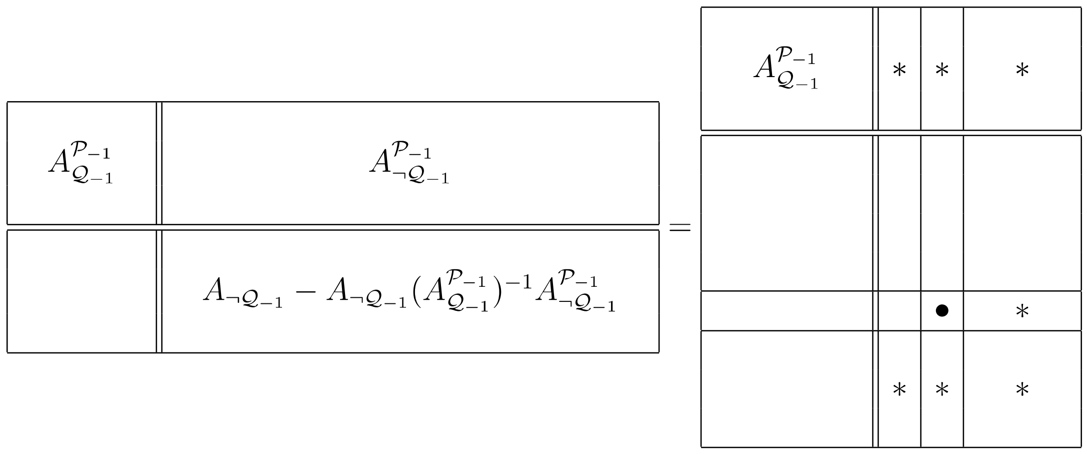
Example 1. *At step s we have the augmented system*



$A^{\mathcal{P}_{-1}}$
A
$=$
$A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}}$
$A^{\mathcal{P}_{-1}}_{\neg\mathcal{Q}_{-1}}$
$*$
$*$
$A^{[i_s]}_{\mathcal{Q}_{-1}}$
$A^{[i_s]}_{\neg\mathcal{Q}_{-1}}$
$*$
$*$

.

After using $A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}}$ to zero the entries below it, we have



$A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}}$
$A^{\mathcal{P}_{-1}}_{\neg\mathcal{Q}_{-1}}$
$A_{\neg\mathcal{Q}_{-1}} - A_{\neg\mathcal{Q}_{-1}}(A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}})^{-1}A^{\mathcal{P}_{-1}}_{\neg\mathcal{Q}_{-1}}$
$=$
$A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}}$
$*$ $*$ $*$
$\bullet$ $*$
$*$ $*$ $*$

Recall that blocks are left blank if they are necessarily 0*. The entry* $\bullet$ *is the first nonzero entry in the first nonzero row, giving indices* i_s *and* j_s*.*

2.3 Useful Properties for a Randomized Algorithm

The idea in [4, 5] to achieve a randomized algorithm with a better runtime compared to the deterministic Algorithm 1 (`RowRankProfile`) is to take a random linear combination $b \in \mathsf{K}^{n\times 1}$ of the columns of $A \in \mathsf{K}^{n\times m}$. Then, the first nonzero row in the Schur complement $A_{\neg\mathcal{Q}} - A_{\mathcal{Q}}(A^{\mathcal{P}}_{\mathcal{Q}})^{-1}A^{\mathcal{P}}_{\neg\mathcal{Q}}$ can be found with high probability by finding the first nonzero entry in the single vector $b - A_{\mathcal{Q}}(A^{\mathcal{P}}_{\mathcal{Q}})^{-1}b^{\mathcal{P}}$.

Let $b \in \mathsf{K}^{n\times 1}$ be sampled randomly and uniformly from the column space of A, i.e., $b = Ag$ for some uniformly and randomly chosen $g \in \mathsf{K}^{m\times 1}$. Consider the following system and

$$\left[\begin{array}{c|c||c} A^{\mathcal{P}}_{\mathcal{Q}} & A^{\mathcal{P}}_{\neg\mathcal{Q}} & b^{\mathcal{P}} \\ \hline A_{\mathcal{Q}} & A_{\neg\mathcal{Q}} & b \end{array}\right].$$

Then the transformation of b equivalent to the transformation of $A_{\neg\mathcal{Q}}$ through the transformation in (2.2) is $b - A_{\mathcal{Q}}(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}b^{\mathcal{P}}$.

$$\left[\begin{array}{c|c||c} A_{\mathcal{Q}}^{\mathcal{P}} & A_{\neg\mathcal{Q}}^{\mathcal{P}} & b^{\mathcal{P}} \\ \hline 0 & A_{\neg\mathcal{Q}} - A_{\mathcal{Q}}(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}A_{\neg\mathcal{Q}}^{\mathcal{P}} & b - A_{\mathcal{Q}}(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}b^{\mathcal{P}} \end{array}\right]. \tag{2.6}$$

The following lemma and theorem are useful results from [4].

Lemma 5. *Let $\mathcal{P} = [i_1, \dots, i_s]$ be a proper prefix of the row rank profile of A. Then if*

$$A_{\neg\mathcal{Q}_{-1}}^{[i_s]} - A_{\mathcal{Q}_{-1}}^{[i_s]}(A_{\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}})^{-1}A_{\neg\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}} \neq 0 \tag{2.7}$$

then

$$b^{[i_s]} - A_{\mathcal{Q}_{-1}}^{[i_s]}(A_{\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}})^{-1}b^{\mathcal{P}_{-1}} \neq 0 \tag{2.8}$$

with probability $(1 - \frac{1}{\#\mathsf{K}})$.

Proof. Let $A_{\neg\mathcal{Q}_{-1}}^{[i_s]} - A_{\mathcal{Q}_{-1}}^{[i_s]}(A_{\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}})^{-1}A_{\neg\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}} \neq 0$. There is only 1 possible value $b^{[i_s]}$ can take within K for which $b^{[i_s]} = A_{\mathcal{Q}}^{[i_s]}(A_{\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}})^{-1}b^{\mathcal{P}_{-1}}$. Therefore, with probability $(1 - \frac{1}{\#\mathsf{K}})$, if $A_{\neg\mathcal{Q}_{-1}}^{[i_s]} - A_{\mathcal{Q}_{-1}}^{[i_s]}(A_{\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}})^{-1}A_{\neg\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}} \neq 0$, then $b^{[i_s]} - A_{\mathcal{Q}_{-1}}^{[i_s]}(A_{\mathcal{Q}_{-1}}^{\mathcal{P}_{-1}})^{-1}b^{\mathcal{P}_{-1}} \neq 0$. ☐

The following theorem follows from Lemma 5 and is paraphrased from [4, Theorem 6].

Theorem 6. *Let $\mathcal{R} = [i_1, \dots, i_r]$ be the row rank profile of A and $\mathcal{Q} = [j_1, \dots, j_r]$ be a list of column indices such that $A_{\mathcal{Q}}^{\mathcal{R}}$ has generic rank profile. Let $\mathcal{R}_s$ and $\mathcal{Q}_s$ denote the prefixes of $\mathcal{R}$ and $\mathcal{Q}$ of length s, $1 \leq s \leq r$. Then all r field elements*

$$b^{[i_1]} - A_{\mathcal{Q}_1}^{[i_1]}(A_{\mathcal{Q}_1}^{\mathcal{R}_1})^{-1}b^{\mathcal{R}_1}, \quad b^{[i_2]} - A_{\mathcal{Q}_2}^{[i_2]}(A_{\mathcal{Q}_2}^{\mathcal{R}_2})^{-1}b^{\mathcal{R}_2}, \quad \dots \quad , b^{[i_r]} - A_{\mathcal{Q}_r}^{[i_r]}(A_{\mathcal{Q}_r}^{\mathcal{R}_r})^{-1}b^{\mathcal{R}_r}$$

are nonzero simultaneously with probability $(1 - \frac{1}{\#\mathsf{K}})^r$.

Algorithm 2 RandomRowRankProfile

```
Input: $A \in \mathsf{K}^{n\times m}$
Output: $\mathcal{P} = [i_1, \dots, i_r]$, the row rank profile of $A$
  $\{B$ stores $(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}$ after each step$\}$
  $\{x$ stores $(A_{\mathcal{Q}}^{\mathcal{P}})^{-1} b^{\mathcal{P}}$ after each step$\}$
  $\{$Note that if $\mathcal{P} = [\ ]$ or $\mathcal{Q} = [\ ]$, the submatrices they denote have a dimension of 0$\}$
  Choose $g \in \mathsf{K}^{m\times 1}$ randomly and uniformly
  Compute $b := Ag$
  Initialize $\mathcal{P}, \mathcal{Q} \leftarrow [\ ]$, $B \leftarrow [\ ] \in \mathsf{K}^{0\times 0}$, $x \leftarrow [\ ] \in \mathsf{K}^{0\times 1}$,
  $s \leftarrow 1$
  for $i$ from 1 to $n$ do
    if $b^{[i]} - A_{\mathcal{Q}}^{[i]} x \neq 0$ then
      $v := A_{\mathcal{Q}}^{[i]}$
      Precompute $vB$
      $t := A_{\neg\mathcal{Q}}^{[i]} - vBA_{\neg\mathcal{Q}}^{\mathcal{P}}$
      Let $j \in \neg\mathcal{Q}$ such that $t_{[j]} \neq 0$ be minimal
      $u := A_{[j]}^{\mathcal{P}}$, $d := A_{[j]}^{[i]}$
      Precompute $Bu$, $vx$ and $w := (d - vBu)^{-1}$
      Update $B \leftarrow \left[\begin{array}{c|c} B + (Bu)w(vB) & -(Bu)w \\ \hline -w(vB) & w \end{array}\right]$
      Update $x \leftarrow \left[\begin{array}{c} x + (Bu)w(vx) - (Bu)wb^{[i_s]} \\ \hline wb^{[i_s]} - wvx \end{array}\right]$
      $i_s \leftarrow i$, $j_s \leftarrow j$
      $\mathcal{P} \leftarrow [\mathcal{P}, i_s]$, $\mathcal{Q} \leftarrow [\mathcal{Q}, j_s]$
      $s \leftarrow s + 1$
    end if
  end for
  return $\mathcal{P}$
```

2.4 Monte Carlo Computation of the Row Rank Profile

Algorithm 2 (`RandomRowRankProfile`) runs exactly the same as Algorithm 1 (`RowRank-Profile`), except we precompute $b := Ag$ and then, to find the next linearly independent row in a Monte Carlo fashion, we iterate through the transformation $b - A_{\mathcal{Q}}(A_{\mathcal{Q}}^{\mathcal{P}})b^{\mathcal{P}}$ of the single vector b instead of the transformation $A_{\neq \mathcal{Q}} - A_{\mathcal{Q}}(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}A_{\neg\mathcal{Q}}^{\mathcal{P}}$ of the entire matrix. A considerable amount of computation is saved since the matrix $(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}A_{\neg\mathcal{Q}}^{\mathcal{P}}$ does not need to be computed.

Theorem 7. *Monte Carlo Algorithm 2 (`RandomRowRankProfile`) computes the row rank profile $\mathcal{P}$ of a matrix $A \in \mathsf{K}^{n\times m}$ with probability $(1 - \frac{1}{\#\mathsf{K}})^r$. The algorithm runs in $mr^2 + \frac{8}{3}r^3 + O(mr + r^2 + |A|)$.*

Proof. The correctness of the algorithm follows from the lemmas. The probability follows from Theorem 6. The time to compute $A_{\mathcal{Q}}^{\mathcal{P}}$ is $2r^3$ overall, and is the dominant cost in the algorithm. From Lemma 2 and Corollary 3, the time to compute B and $Bb^{\mathcal{P}}$ is

$$\sum_{s=1}^{r}(6s^2 + O(s)) = 2r^3 + O(r^2).$$

The cost of computing $A_{\mathcal{Q}}^{[i]}BA_{\neg\mathcal{Q}}^{\mathcal{P}}$ for a given step s is $2s^2 + O(s)$ operations for $A_{\mathcal{Q}}^{[i]}B$, and $2ms - 2s^2 + O(m+s)$ operations for $A_{\mathcal{Q}}^{[i]}BA_{\neg\mathcal{Q}}^{\mathcal{P}}$. Subtracting the result from $A^{[i]}$ is an $O(s)$ operation.

$$\sum_{s=1}^{r}(2ms + O(m+s)) = mr^2 + O(mr + r^2)$$

While vB is precomputed as $A_{\mathcal{Q}}^{[i]}B$, the cost of computing Bu at every step s is $2s^2 + O(s)$. This gives the following overall cost for computing Bu:

$$\sum_{s=1}^{r}(2s^2 + O(s)) = \frac{2}{3}r^3 + O(r^2).$$

Since i iterates from 1 to n with no back tracking through the entire algorithm, and since $(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}$ is precomputed, the algorithm will access each nonzero element of A at most twice, while ignoring the zeroes. Thus, we have a total runtime of $O(|A|)$ to compute each $b^{[i]} - A_{\mathcal{Q}}^{[i]}x$.

So the total runtime of the algorithm is $mr^2 + \frac{8}{3}r^3 + O(mr + r^2 + |A|)$. □

Example 2. *At step s we have the augmented system:*

$$\left[\begin{array}{c||c} A^{\mathcal{P}_{-1}} & b^{\mathcal{P}_{-1}} \\ \hline A & b \end{array}\right] = \left[\begin{array}{c|c||c} A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}} & A^{\mathcal{P}_{-1}}_{\neg\mathcal{Q}_{-1}} & b^{\mathcal{P}_{-1}} \\ \hline * & * & * \\ \hline A^{[i_s]}_{\mathcal{Q}_{-1}} & A^{[i_s]}_{\neg\mathcal{Q}_{-1}} & b^{[i_s]} \\ \hline * & * & * \end{array}\right]$$

After reduction we have

$$\left[\begin{array}{c||c||c} A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}} & A^{\mathcal{P}_{-1}}_{\neg\mathcal{Q}_{-1}} & b^{\mathcal{P}_{-1}} \\ \hline & A_{\neg\mathcal{Q}_{-1}} - A_{\mathcal{Q}_{-1}}(A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}})^{-1}A^{\mathcal{P}_{-1}}_{\neg\mathcal{Q}_{-1}} & b - A_{\mathcal{Q}_{-1}}(A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}})^{-1}b^{\mathcal{P}_{-1}} \end{array}\right] =$$

$$\left[\begin{array}{c||c|c|c||c} A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}} & * & * & * & b^{\mathcal{P}_{-1}} \\ \hline & * & * & * & \\ \hline & & \bullet & * & \bullet \\ \hline & * & * & * & * \end{array}\right]$$

Recall that $*$ indicates a possible nonzero. The row index of the nonzero in $b - A_{\mathcal{Q}_{-1}}(A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}})^{-1}b^{\mathcal{P}_{-1}}$ gives index i_s, The column index j_s is found by computing $A^{[i_s]}_{\neg\mathcal{Q}_{-1}} -$

$A^{[i_s]}_{\mathcal{Q}_{-1}}(A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}})^{-1}A^{\mathcal{P}_{-1}}_{\neg\mathcal{Q}_{-1}}$. *Note that there may still be nonzero rows before* i_s *that are missed because their respective entries in* $b - A_{\mathcal{Q}_{-1}}(A^{\mathcal{P}_{-1}}_{\mathcal{Q}_{-1}})^{-1}b^{\mathcal{P}_{-1}}$ *are zero.*

2.5 Handling Cases with Low Probability of correctness

First note that the probability of correctness of Algorithm 2 (`RandomRowRankProfile`) depends on the size of the field and rank of the input matrix. As such, there will be certain cases where the algorithm is more likely to fail than succeed. For example if $\mathsf{K} = \mathbb{Z}_3$ and A has a rank of 4, then the probability that $\mathcal{P}$ is actually the row rank profile of A is 0.198.

The chance of correctness can be improved in two ways at the expense of runtime. Let p be a positive integer parameter. The first method is to simply run the algorithm p times, generating a new $g \in \mathsf{K}^{m\times 1}$ each time. We call this method `MultiRun`(A, p). The second method is to generate p vectors for b instead of a single vector, and use them all in a single run. Then, when checking if $b^{[i]} - A^{[i]}_{\mathcal{Q}}x = 0$, all p samples are checked. We call this method `MultiColumn`(A, p).

`MultiRun(A, p)` Run `RandomRowRankProfile` p times. Return the lexicographically minimal computed $\mathcal{P}$ that is of the longest length.

`MultiColumn(A, p)` Run `RandomRowRankProfile` with $g \in \mathsf{K}^{m\times p}$, checking that all p elements of a row give a nonzero before declaring the row linearly dependent. Return $\mathcal{P}$ as computed this way.

We will show that algorithm `MultiColumn`(A, p) is the better approach. Algorithm `MultiColumn`(A, p) runs faster and is more likely to produce the correct result compared to `MultiRun`(A, p). Moreover, if a lower bound on the probability of correctness is desired, a smaller lower bound on parameter p is required by `MultiColumn`(A, p) to ensure the desired lower bound on the probability.

2.5.1 Improvement Through Multiple Runs

First consider Algorithm `MultiRun`(A, p).

Theorem 8. *Let $\mathcal{P}$ be the result from Algorithm* ***MultiRun***(A, p)*. The probability that $\mathcal{P}$ is the row rank profile of A is $1-(1-(1-\frac{1}{\#\mathsf{K}})^r)^p$ and the algorithm runs in $p(mr^2 + \frac{8}{3}r^3 + O(mr + r^2 + |A|))$ field operations.*

Proof. The runtime is correct since the modification simply involves running `RandomRowRankProfile` p times. The correctness of the probability follows from the independence of successive runs. Each run has a probability of correctness of $(1-\frac{1}{\#\mathsf{K}})^r$, from Theorem 7. Thus each run, independently, has a probability of failure of $1-(1-\frac{1}{\#\mathsf{K}})^r$. Thus, the probability that at least one of the p runs succeeds $1-(1-(1-\frac{1}{\#\mathsf{K}})^r)^p$. □

Given the probability of correctness in Theorem 8, it is interesting to find a lower bound for p to ensure the probability of correctness of `MultiRun`(A, p) is at least 1/2. By setting the probability of correctness to be 1/2, and solving for p, we obtain the following corollary. Note that if the rank is not known, $\min(n, m)$ can be substituted for r since this can only result in making p larger than necessary.

Corollary 9. *If*

$$p > \frac{-1}{\log_2(1-(1-\frac{1}{\#\mathsf{K}})^r)}$$

then Algorithm ***MultiRun***(A, p) *will produce the correct result with probability at least* $1/2$.

Example 3. *Suppose that while running* ***MultiRun***(A, p)*, on 2 distinct runs q_1 and q_2,* ***RandomRowRankProfile*** *is running at some step s, and $\mathcal{P}$ is correct thus far. Let b_{q_1}, b_{q_2}, where $1 \le q_1, q_2 \le p$, denote 2 distinct samples from the column space of A. These vectors are the b vectors from trials q_1 and q_2. Suppose $b_{q_1} - A_{\mathcal{Q}}x_{q_1}$ and $b_{q_2} - A_{\mathcal{Q}}x_{q_2}$ are as follows*

$$b_{q_1} - A_{\mathcal{Q}}x_{q_1} = \begin{array}{|c|} \hline 0 \\ 0 \\ 0 \\ \bullet \\ * \\ * \\ * \\ * \\ \hline \end{array}, b_{q_2} - A_{\mathcal{Q}}x_{q_2} = \begin{array}{|c|} \hline 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \bullet \\ * \\ * \\ \hline \end{array}$$

On trial q_1, $i_s = 4$ will be identified, while on trial q_2, i_s will be misidentified as row 6, illustrating how the probability of correctness is increased - it is unlikely that several *samples will all be bad. However, the profile found in trial q_2 is necessarily going to be erroneous, since no information is shared between 2 runs.*

2.5.2 Improvement Through Multiple Column Samples

Now, consider Algorithm `MultiColumn`(A, p), running the algorithm with more vectors b, i.e., let $b \in \mathsf{K}^{n\times p}$ rather than $b \in \mathsf{K}^{n\times 1}$.

Theorem 10. *Algorithm `MultiColumn`(A, p) returns the correct result with probability $(1-(\frac{1}{\#\mathsf{K}})^p)^r$. The running time of the algorithm is $mr^2 + (2p + \frac{2}{3})r^3 + O(pmr + pr^2 + p|A|)$ field operations.*

Proof. The run time follows from the cost of updating x and computing $A_Q^{[i]}$, which are increased by a factor of p times greater than it was originally. The rest of the algorithm has an identical runtime since we only create and update one instance of B, which has the same size and complexity as in the original algorithm. The probability of correctness follows from the independence of each column of b. From Lemma 5, the probability that any given column of b has a false zero in row i is $1/\#\mathsf{K}$. So by the independence of each choice of column, the probability that all columns have false zeroes on a given row is $(1/\#\mathsf{K})^p$. Thus the probability that the algorithm successfully returns the row rank profile of A is $(1 - (\frac{1}{\#\mathsf{K}})^p)^r$. □

Thus, as in Corollary 9, the probability of correctness is at least 1/2 if the following lower bound for p holds. It can be derived by solving for p. As in Corollary 9, substitute $\min(m, n)$ for r if the rank is not known, as this can only make the algorithm more likely to succeed.

Corollary 11. *If*

$$p > \frac{-\log_2(1 - (\frac{1}{2})^{\frac{1}{r}})}{\log_2(\#\mathsf{K})}$$

then Algorithm `MultiColumn`(A, p) will succeed with probability at least 1/2*.*

Note that the `MultiColumn`(A, p) is both faster and succeeds with higher probability for a given p. Moreover, `MultiColumn`(A, p) has a smaller lower bound on p to guarantee a certain probability of correctness. However, the `MultiRun`(A, p) is more readily parallelized, since the runs can simply be run without requiring that different processes communicate during execution

Example 4. *Suppose, using* ***MultiColumn****(A,p),* ***RandomRowRankProfile*** *is running at some step s, and $\mathcal{P}$ is correct thus far. Let b_{q_1}, b_{q_2}, where $1 \le q_1, q_2 \le p$ denote 2 distinct samples from the column space of A. These vectors are columns q_1 and q_2 of b. Suppose $b_{[q_1,q_2]} - A_{\mathcal{Q}}x_{[q_1,q_2]}$ is as follows*

$$b_{[q_1,q_2]} - A_{\mathcal{Q}}x_{[q_1,q_2]} = \boxed{\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \bullet & 0 \\ * & 0 \\ * & \bullet \\ * & * \\ * & * \end{array}}$$

Note that by considering all samples simultaneously, i_s will be found to be 4 despite the fact that column q_2 has a false zero at row 4. Since q_1 does not have a false zero, the row will not be dismissed as linearly dependent. Using multiple samples to mutually confirm whether a row is linearly dependent allows certain samples to occasionally give poor results without compromising the output of the algorithm, ensuring that the result is more likely to be correct.

2.5.3 Comparing the Methods of Improvement

For a fixed p, the `MultiColumn`(A,p) has a better runtime than `MultiRun`(A,p). Furthermore, if a lower bound on the probability of correctness is desired, the lower bound on p given by Corollary 11 for `MultiColumn`(A,p) is considerably smaller than the bound given by Corollary 9 for `MultiRun`(A,p). Thus, `MultiColumn`(A,p) is the better approach for increasing the probability of correctness of Algorithm 2 (`RandomRowRankProfile`).

Example 5. *The Table below illustrates the lower bound on p required for each method to guarantee that Algorithm 2 (****RandomRowRankProfile****) succeeds with probability at least $1/2$ for various matrices A.*

Field K	*rank(A)*	*Correctness Prob.*	`MultiRun` *bound*	`MultiColumn` *bound*
$\mathbb{Z}_2$	5	0.0312	22	3
$\mathbb{Z}_2$	11	0.00049	1420	5
$\mathbb{Z}_3$	5	0.1317	5	2
$\mathbb{Z}_3$	11	0.0116	60	3
$\mathbb{Z}_{101}$	120	0.3030	2	2
$\mathbb{Z}_{101}$	200	0.1367	5	2
$\mathbb{Z}_{151}$	120	0.4505	2	2
$\mathbb{Z}_{151}$	200	0.2648	3	2

Note that the lower bound on p is consistently much smaller if using `MultiColumn`(A, p) *than if using* `MultiRun`(A, p). *This is because the correctness of* `MultiRun`(A, p) *depends on generating some correct vector $g \in \mathsf{K}^{m\times 1}$ after p trials, while* `MultiColumn`(A, p) *can use information from all p samples to better determine linear independence of a row.*

Chapter 3

Row Rank Profile Certification

We now present a Monte Carlo certificate to verify the accuracy of a given row rank profile $\mathcal{P}$. The algorithm runs in $4r^2 + O(r + |A|)$. If the result is FALSE, then $\mathcal{P}$ is not the row rank profile of A. If $\mathcal{P}$ is the row rank profile of A, the algorithm returns TRUE with certainty. If $\mathcal{P}$ is not the row rank profile of A, the probability that TRUE is incorrectly reported is bounded by $2/\#\mathsf{K}$.

3.1 Inverse Decomposition

The following Lemma, from [5, Section 7.1], is useful for storing a decomposition of the inverse of $A_{\mathcal{Q}}^{\mathcal{P}}$ and all its leading submatrices. If $A_{\mathcal{Q}}^{\mathcal{P}}$ has generic rank profile, then $(A_{\mathcal{Q}}^{\mathcal{P}})^{-1}$ has a unique decomposition $(A_{\mathcal{Q}}^{\mathcal{P}})^{-1} = (R_r L_r) \cdots (R_1 L_1)$.

Lemma 12. *Let $A_{\mathcal{Q}}^{\mathcal{P}}$ have generic rank profile and $r = rank(A)$. For any s such that $1 \le s \le r$, the inverse of the leading $s \times s$ submatrix of $A_{\mathcal{Q}}^{\mathcal{P}}$ is the leading $s \times s$ submatrix of the product of unique factors $(R_s L_s)(R_{s-1} L_{s-1}) \cdots (R_1 L_1)$, where*

$$R_i = \left[\begin{array}{c|c|c} & * & \\ I_{i-1} & * & \\ & * & \\ \hline & * & \\ \hline & & I_{r-i} \end{array}\right] \in \mathsf{K}^{r \times r}, L_i = \left[\begin{array}{ccc|c|c} & & & & \\ & I_{i-1} & & & \\ & & & & \\ \hline * & * & * & 1 & \\ \hline & & & & I_{r-i} \end{array}\right] \in \mathsf{K}^{r \times r}.$$

Thus all r of the inverses of the leading submatrices of $A_{\mathcal{Q}}^{\mathcal{P}}$ can be stored in $\Theta(r^2)$ space.

3.2 Monte Carlo Certificate of Row Rank Profile

Certification requires the input matrix $A \in \mathsf{K}^{n\times m}$, the claimed solution $\mathcal{P}$ and a certificate $(\mathcal{Q}, ((L_r, R_r), (L_{r-1}, R_{r-1}), \cdots, (L_1, R_1)))$, where $\mathcal{P}$ and $\mathcal{Q}$ are lists of row and column indices respectively, and $(R_r, L_r), \cdots, (R_1, L_1)$ claimed to be the decomposition of the inverse of $A^{\mathcal{P}}_{\mathcal{Q}}$. The correctness of $\mathcal{P}$ as the row rank profile of A can be certified using Algorithm 3 (`RandomRowProfileCertificate`). The algorithm proceeds in two steps. First, certify that $A^{\mathcal{P}}_{\mathcal{Q}}$ is invertible, and that the inverse is $(R_rL_r)\cdots(R_1L_1)$. It then follows that $A^{\mathcal{P}}_{\mathcal{Q}}$ has generic rank profile. Thus $\mathcal{P}$ is at least a set of linearly independent rows, required of the row rank profile, so failing this test causes FALSE to be reported. Second, Algorithm 2 (`RandomRowRankProfile`) is modified to run using already computed values for $\mathcal{P}$ and $\mathcal{Q}$ and $(A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$ to test the claimed linear dependence of rows in $\neg\mathcal{P}$. If, at any point, the algorithm finds a row index $i \in \neg\mathcal{P}$ that should be in $\mathcal{P}$, `RandomRowProfileCertificate` terminates and FALSE is reported since the row rank profile must be lexicographically minimal and have length rank(A). If all tests are passed, then `RandomRowProfileCertificate` returns TRUE. However if $\mathcal{P}$ is not the row rank profile of A, then the algorithm will incorrectly return TRUE with probability at most $2/\#\mathsf{K}$.

The analysis of Algorithm 3 (`RandomRowProfileCertificate`) requires some lemmas to establish the probability of correctness.

Given $A^{\mathcal{P}}_{\mathcal{Q}}$ and $(R_rL_r)(R_{r-1}L_{r-1})\cdots(R_1L_1)$, we can certify that the decomposition $(R_rL_r)(R_{r-1}L_{r-1})\cdots(R_1L_1)$ is the inverse of $A^{\mathcal{P}}_{\mathcal{Q}}$ in a Monte Carlo fashion as follows: for some randomly and uniformly chosen $c \in \mathsf{K}^{r\times 1}$, if $BA^{\mathcal{P}}_{\mathcal{Q}}c \neq c$, then B is not the inverse of $A^{\mathcal{P}}_{\mathcal{Q}}$. The following Lemma is an adaptation of Frievalds' algorithm for certifying the product of 2 matrices.

Lemma 13. *Let $A^{\mathcal{P}}_{\mathcal{Q}}$, $B := (R_rL_r)\cdots(R_1L_1) \in \mathsf{K}^{r\times r}$ and $c \in \mathsf{K}^{r\times 1}$ be chosen uniformly and randomly. If $A^{\mathcal{P}}_{\mathcal{Q}}$ and its leading submatrices are invertible, then $BA^{\mathcal{P}}_{\mathcal{Q}}c = c$ with certainty. If B is not the inverse of $A^{\mathcal{P}}_{\mathcal{Q}}$, then $BA^{\mathcal{P}}_{\mathcal{Q}}c = c$ with probability at most $1/\#\mathsf{K}$.*

Proof. Let $BA^{\mathcal{P}}_{\mathcal{Q}}c = c$. First consider the case when $B = (A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$. Then $BA^{\mathcal{P}}_{\mathcal{Q}}c = c$ with certainty. So we need to consider the case when $B \neq (A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$. So $(BA^{\mathcal{P}}_{\mathcal{Q}} - I)c = 0$, but $(BA^{\mathcal{P}}_{\mathcal{Q}} - I) \neq 0$. As in Lemma 5, $(BA^{\mathcal{P}}_{\mathcal{Q}} - I)^{[i]}c = 0$, for some nonzero row i, with probability $1/\#\mathsf{K}$. Since $BA^{\mathcal{P}}_{\mathcal{Q}} - I \neq 0$, there must be at least one row of $BA^{\mathcal{P}}_{\mathcal{Q}} - I$ which is nonzero. Then $BA^{\mathcal{P}}_{\mathcal{Q}}c = c$ with probability $1/\#\mathsf{K}$. So when $B \neq (A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$, then $BA^{\mathcal{P}}_{\mathcal{Q}}c = c$ with probability at most $1/\#\mathsf{K}$. Moreover, if $A^{\mathcal{P}}_{\mathcal{Q}}c \neq c$ then $B \neq (A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$ with certainty. ☐

Note that if there are p nonzero, linearly independent rows in $(R_rL_r)\cdots(R_1L_1)A^{\mathcal{P}}_{\mathcal{Q}} - I$, then $(R_rL_r)\cdots(R_1L_1)A^{\mathcal{P}}_{\mathcal{Q}}c = c$ with probability $(1/\#\mathsf{K})^p$, which is much smaller than

Algorithm 3 RandomRowProfileCertificate

```
Input: $A \in \mathsf{K}^{n\times m}$, $\mathcal{P}$, $\mathcal{Q}$, and $((R_r, L_r), (R_{r-1}, L_{r-1}), \cdots, (R_1, L_1))$
Output: FALSE if $\mathcal{P}$ is shown to be incorrect, TRUE otherwise
  {$\mathcal{P}_s$ and $\mathcal{Q}_s$ are prefixes of length $s$ of $\mathcal{P}$ and $\mathcal{Q}$}
  {$x$ stores $(A_{\mathcal{Q}_s}^{\mathcal{P}_s})^{-1} b^{\mathcal{P}_s}$ for each step}
  Choose $c \in \mathsf{K}^{r\times 1}$ randomly and uniformly
  if $((R_r, L_r), (R_{r-1}, L_{r-1}), \cdots, (R_1, L_1)) A_{\mathcal{Q}}^{\mathcal{P}} c \neq c$ then
    return FALSE
  end if
  Choose $g \in \mathsf{K}^{m\times 1}$ randomly and uniformly
  Compute $b := Ag$
  Initialize $x \leftarrow [\ ] \in \mathsf{K}^{0\times 1}$
  $s \leftarrow 0$
  for $i$ from 1 to $n$ do
    if $i \in \mathcal{P}$ then
      $s \leftarrow s + 1$
      Update $x \leftarrow R_s L_s \begin{bmatrix} x \\ b^{[i]} \end{bmatrix}$
    else if $b^{[i]} - A_{\mathcal{Q}_s}^{[i]} x \neq 0$ then
      return FALSE
    end if
  end for
  return TRUE
```

$1/\#\mathsf{K}$. Thus the estimate in Lemma 13 is a worst case upper bound. Also note that if $(R_rL_r)\cdots(R_1L_1) = (A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$, then $A^{\mathcal{P}}$ has full row rank, and rank$(A) \geq |\mathcal{P}|$.

Lemma 14. *Assume $(R_rL_r)\cdots(R_1L_1) = (A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$. If $\mathcal{P}$ is the rank profile of A then Algorithm 3 (`RowRankProfileCertificate`) returns TRUE.*

We now try to bound the probability that an incorrect $\mathcal{P}$ misses detection in the algorithm.

Lemma 15. *Assume $(R_rL_r)\cdots(R_1L_1) = (A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$ and let $b \in \mathsf{K}^{n\times 1}$ be uniformly and randomly sampled from the column space of A. Suppose there exists an $i \in \neg\mathcal{P}$ such that $A^{[i]}$ is linearly independent of rows $A^{\mathcal{P}_s}$, where s is such that $i_s \leq i \leq i_{s+1}$ with $i_0 = 0$ and $i_{r+1} = n+1$. Then $b^{[i]} - A^{[i]}_{\neg\mathcal{Q}_s}(A^{\mathcal{P}_s}_{\mathcal{Q}_s})^{-1}b^{\mathcal{P}_s} = 0$ with probability $1/\#\mathsf{K}$.*

Proof. Note that if $i \in \neg\mathcal{P}$ is the index of a row that is linearly independent from the rows in $A^{\mathcal{P}_s}$, then the row $A^{[i]}_{\neg\mathcal{Q}_s} - A^{[i]}_{\mathcal{Q}_s}(A^{\mathcal{P}_s}_{\mathcal{Q}_s})^{-1}A^{\mathcal{P}_s}_{\neg\mathcal{Q}_s} \neq 0$. Thus, as with Lemma 5, the probability that $b^{[i]} - A^{[i]}_{\neg\mathcal{Q}_s}(A^{\mathcal{P}_s}_{\mathcal{Q}_s})^{-1}b^{\mathcal{P}_s} = 0$ is $1/\#\mathsf{K}$ □

Lemma 16. *Assume $(R_rL_r)\cdots(R_1L_1) = (A^{\mathcal{P}}_{\mathcal{Q}})^{-1}$ but $\mathcal{P}$ is not the rank profile of A. Then there exists an $i \notin \mathcal{P}$ such that row $A^{[i]}$ is linearly independent of rows $A^{\mathcal{P}_s}$.*

We now establish the runtime and probability of correctness of Algorithm 3 (`Random-RowProfileCertificate`) using Lemmas 13 and 15. Recall that the algorithm takes as input a matrix $A \in \mathsf{K}^{n\times m}$, a claimed solution $\mathcal{P}$ and a certificate $\mathcal{Q}, ((R_r, L_r), \cdots, (R_1, L_1)))$, with the claim that $A^{\mathcal{P}}_{\mathcal{Q}}$ has full rank and generic rank profile, that $(R_rL_r)\cdots(R_1L_1)$ is its inverse, that $\mathcal{P}$ is lexicographically minimal and of maximal length.

Theorem 17. *Monte Carlo Algorithm 3 (`RandomRowProfileCertificate`) has the following properties:*

- *If FALSE is returned, then $\mathcal{P}$ is not the row rank profile of A.*
- *If $\mathcal{P}$ is the row rank profile of A, TRUE returned.*
- *If $\mathcal{P}$ is not the row rank profile of A, then with probability at most $2/\#\mathsf{K}$, TRUE is incorrectly returned.*

The algorithm runs in $4r^2 + O(r + |A|)$ field operations from K.

Proof. The correctness of Algorithm 3 (`RowRankProfileCertificate`) and the probability that TRUE is incorrectly returned follows from Lemmas 13, 14, 15 and 16. If $B \neq A_{\mathcal{Q}}^{\mathcal{P}}$, the probability that the algorithm will fail to detect this is at most $1/\#\mathsf{K}$. The probability that $\mathcal{P}$ incorrectly appears to be the correct row rank profile of A by the algorithm is $1/\#\mathsf{K}$. Combining these gives a total probability of having a false positive returned bounded by $2/\#\mathsf{K}$. Note that if $\mathcal{P}$ is the row rank profile, then it must pass the tests.

We now establish the runtime. First, the vector $(L_rR_r)\cdots(L_1R_1)A_{\mathcal{Q}}^{\mathcal{P}}c$ is computed. $O(|A_{\mathcal{Q}}^{\mathcal{P}}|)$ field operations from K are required to compute $A_{\mathcal{Q}}^{\mathcal{P}}c$. The subsequent r sets of two products require $4s-4$ field operations for each s such that $1 \leq s \leq r$, since the sparsity structure of L_s and R_s is exploited. This gives a time of

$$O(|A_{\mathcal{Q}}^{\mathcal{P}}|) + \sum_{s=1}^{r}(4s-4) = 2r^2 + O(r + |A_{\mathcal{Q}}^{\mathcal{P}}|).$$

Second, Algorithm 3 (`RowRankProfileCertificate`) requires r updates to $x = Bb^{\mathcal{P}}$ each requiring $4s-2$ field operations from K for each step s from 1 to r. This gives a time of

$$\sum_{s=1}^{r}(4s-2) = 2r^2 + O(r).$$

Similarly, there are $n-r$ tests for each row in $\neg\mathcal{P}$. Running all of these tests takes $O(|A^{\neg\mathcal{P}}|)$ since it involves $n-r$ dot products with the rows in $\neg\mathcal{P}$.

This gives a total runtime of $4r^2 + O(r + |A|)$. □

3.3 Handling Cases with Low Probability of Correctness

Similar to Algorithm 2 (`RandomRowRankProfile`), the probability of correctness of Algorithm 3 (`RandomRowProfileCertificate`) depends on the size of K. As such, a sufficiently small field gives a high probability of incorrectly certifying that the given list $\mathcal{P}$ is the row rank profile. As in the case of Algorithm 2 (`RandomRowRankProfile`) we can increase the likelihood of correctness by running the algorithm p times, or by taking p samples from the column space of A, i.e., $g \in \mathsf{K}^{m\times p} \Rightarrow b := Ag \in \mathsf{K}^{n\times p}$. In either case, we show in the following 2 theorems that the probability TRUE is incorrectly returned is bounded by $2/(\#\mathsf{K})^p$. Both methods result in the same cost increase as well. Using multiple runs, the runtime is increased by a factor of p over the cost the original certification, i.e., $p(4r^2 + O(r + |A|))$. Similarly, if using one run with several samples, the cost of computing $BA^{\mathcal{P}}_{\mathcal{Q}}c$, updating x and multiplication $A^{[i]}_{\neg\mathcal{Q}}x$ will increase by a factor of p more giving a total runtime of $p(4r^2 + O(r + |A|))$.

Theorem 18. *If `MultiRun` is used to run Algorithm 3 (`RandomRowProfileCertificate`) p times, then the probability that the algorithm incorrectly reports that $\mathcal{P}$ is the row rank profile of A p times is at most $2/(\#\mathsf{K})^p$. The runtime is $p(4r^2 + O(r + |A|))$.*

Proof. For `MultiRun` to report a false positive p times, then all tests must give false positives independently. Both of the tests fail to detect an incorrect profile with probability $1/\#\mathsf{K}$ for each run. Thus, each test in the algorithm will incorrectly return TRUE p times with probability $1/(\#\mathsf{K})^p$. This gives an overall probability $2/(\#\mathsf{K})^p$. The runtime is simply p times the runtime of Algorithm 3 (`RandomRowProfileCertificate`). ☐

Theorem 19. *If `MultiColumn` is used to run Algorithm 3 (`RandomRowProfileCertificate`) with p independent samples, i.e. letting $c \in \mathsf{K}^{r\times p}$ and $g \in \mathsf{K}^{m\times p}$, then the probability that the algorithm incorrectly reports that $\mathcal{P}$ is the row rank profile of A is at most $2/(\#\mathsf{K})^p$. The runtime is $p(4r^2 + O(r + |A|))$.*

Proof. Each step of Algorithm 3 (`RandomRowProfileCertificate`) would have to report a false positive p times before moving on to the next step. The probability of this happening for any given step is at most $(1/\#\mathsf{K})^p$ since each report would be independent. Thus the overall probability that TRUE is incorrectly returned is $2/(\#\mathsf{K})^p$. Moreover, the cost of the algorithm comes from updating x and testing $b^{[i]} - A^{[i]}_{\mathcal{Q}_s} \neq 0$. These updates take p times longer than if c and x were individual vectors, so the runtime is increased by a factor of p, giving the total runtime of $p(4r^2 + O(r + |A|))$. ☐

Note that `MultiRun` and `MultiColumn` both yield identical runtimes and probabilities of correctness. Knowing this, we can derive the a bound for p for either method to guarantee that the probability that the algorithm incorrectly returns TRUE when $\mathcal{P} \neq \mathcal{R}$ is at most $1/2$. This bound can be derived by solving for p.

Corollary 20. *Suppose $\mathcal{P}$ is not the row rank profile of A. If*

$$p > \frac{2}{\log_2(\mathsf{K})}$$

then the algorithm will incorrectly certify that $\mathcal{P}$ is the row rank profile of A with probability at most $1/2$.

Chapter 4

Conclusion and Future Work

We've shown two algorithms that operate over a finite field K. The algorithms can work on any field F by selecting a sufficiently large subset $\mathsf{K} \subset \mathsf{F}$ to use. Algorithm 2 (`RandomRowRankProfile`) and Algorithm 3 (`RandomRowProfileCertificate`) work for any field of numbers. Algorithm 2 (`RandomRowRankProfile`) computes the row rank profile of $A \in \mathsf{K}^{n \times m}$ in a Monte Carlo fashion in $mr^2 + \frac{8}{3}r^3 + O(mr + r^2 + |A|)$ field operations from K, where $r = \text{rank}(A)$ and $|A|$ is the number of nonzero entries in A. With probability at least $(1 - \frac{1}{\#\mathsf{K}})^r$, the list returned by `RandomRowRankProfile` is the row rank profile of A. As compared to `ImprovedRankProfile` in [4], `RandomRowRankProfile` has no logarithmic factors in the run time and does not require field extensions to increase the likelihood of correctness. It would be interesting to improve `RandomRowRankProfile` to reduce the runtime's dependence on m.

Algorithm 3 (`RandomRowProfileCertificate`) can certify that a claimed row profile $\mathcal{P}$ of A is correct in a Monte Carlo fashion in $4r^2 + O(r + |A|)$ field operations from K.

Bibliography

[1] M. Nehring E. Kaltofen and B. Saunders. Quadratic-time certificates in linear algebra. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'11*, pages 171–176. ACM Press, New York, 2011.

[2] R. Freivalds. Probabilistic machines can use less running time. In *IFIP Congress*, volume 839, page 842, 1977.

[3] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27:701–717, 1980.

[4] A. Storjohan and S. Yang. Linear independence oracles and applications to rectangular and low rank linear systems. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'14*. ACM Press, New York, 2014.

[5] S. Yang. Algorithms for fast linear system solving and rank profile computation. Master's thesis, University of Waterloo, 2014.

[6] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. EUROSAM 79*, pages 216–226, Marseille, 1979.