Computing a Basis for an Integer Lattice: A Special Case

Haomin Li
Cheriton School of Computer Science
University of Waterloo
h439li@uwaterloo.ca

Arne Storjohann
Cheriton School of Computer Science
University of Waterloo
astorjoh@uwaterloo.ca

ABSTRACT

Consider an integer matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ that has full column rank $n-1$. The set of all $\mathbb{Z}$-linear combinations of the rows of $\boldsymbol{A}$ generates a lattice, denoted by $\mathcal{L}(\boldsymbol{A})$. An algorithm is given that computes a matrix $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ such that $\boldsymbol{B} := \boldsymbol{C}\boldsymbol{A} \in \mathbb{Z}^{(n-1)\times(n-1)}$ is a basis for $\boldsymbol{A}$, that is, $\boldsymbol{B}$ has full row rank $n-1$ and $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{A})$. The matrix $\boldsymbol{C}$ will satisfy $\log ||\boldsymbol{C}|| \leq 4\log n + 2\log ||\boldsymbol{A}||$ (where $||\cdot||$ denotes the maximum entry in absolute value) and will have at most $n(1+\log_2 n)$ nonzero entries. The cost of the algorithm is about the same as that required to multiply together two square integer matrices of dimension n that have magnitude of entries bounded by $||\boldsymbol{A}||$.

CCS CONCEPTS

• **Computing methodologies** → **Linear algebra algorithms**; **Exact arithmetic algorithms**.

KEYWORDS

integer matrix, lattice basis, exact arithmetic algorithm

ACM Reference Format:
Haomin Li and Arne Storjohann. 2022. Computing a Basis for an Integer Lattice: A Special Case. In *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation (ISSAC '22), July 4–7, 2022, Villeneuve-d'Ascq, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3476446.3536184

1 INTRODUCTION

The set of all $\mathbb{Z}$-linear combinations of the rows of an integer matrix $\boldsymbol{A}$ is a (row) lattice, denoted by $\mathcal{L}(\boldsymbol{A})$. We consider the problem of computing a basis for $\mathcal{L}(\boldsymbol{A})$. The most general version of the problem takes as input an $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, and asks as output a $\boldsymbol{B} \in \mathbb{Z}^{r\times m}$ such that $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{A})$, where r is the rank of $\boldsymbol{A}$. Such a $\boldsymbol{B}$ is a basis for $\mathcal{L}(\boldsymbol{A})$. A constraint is that we want the bitlength of entries in the basis to be not much larger than those of $\boldsymbol{A}$. Formally, we require that $\log ||\boldsymbol{B}|| \in O(\log n + \log ||\boldsymbol{A}||)$, where $||\cdot||$ for a matrix or vector denotes the largest entry in absolute value.

Before continuing, we mention that $\mathcal{L}(\boldsymbol{A})$ will always denote the lattice generated by the *rows* of $\boldsymbol{A}$, and, for brevity, we will write "$\boldsymbol{B}$ is a basis for $\boldsymbol{A}$" to mean "$\boldsymbol{B}$ is a basis for $\mathcal{L}(\boldsymbol{A})$."

ISSAC '22, July 4–7, 2022, Villeneuve-d'Ascq, France

ACM ISBN 978-1-4503-8688-3/22/07…$15.00
https://doi.org/10.1145/3476446.3536184

One of our motivations for studying this problem is to extend fast algorithms for computing normal forms of integer matrices to input matrices with arbitrary shape and rank profile. For example, the fastest algorithm for computing the Smith form of an integer matrix by Birmpilis et al. [2] requires as input a square nonsingular input matrix.

By first producing a basis $\boldsymbol{B} \in \mathbb{Z}^{r\times m}$ for $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, then producing a basis $\bar{\boldsymbol{B}} \in \mathbb{Z}^{r\times r}$ for $\text{Transpose}(\boldsymbol{B}) \in \mathbb{Z}^{m\times r}$, we can apply the Smith form algorithm with the nonsingular input $\bar{\boldsymbol{B}}$ to obtain the Smith form of $\boldsymbol{A}$. We refer to Li and Nguyen [6] for other applications of computing a lattice basis with entries not much larger than those of the input matrix, and for a survey of previous approaches for solving this problem.

A classical case of the problem is $n = 2$ and $m = 1$. Computing a basis for

$$\boldsymbol{A} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \in \mathbb{Z}^{2\times 1}$$

corresponds to computing $g = \gcd(a_1, a_2)$. The extended gcd problem asks also for a lattice generator $\boldsymbol{C} \in \mathbb{Z}^{1\times 2}$ such that

$$\boldsymbol{C}\boldsymbol{A} = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} g \end{bmatrix}.$$

We will use a cost function B such that the extended gcd problem can be solved in $O(\mathrm{B}(\log ||\boldsymbol{A}||))$ bit operations, where $||\cdot||$ denotes the maximal entry in absolute value.

Another case is n arbitrary and $m = 1$. Given $\boldsymbol{A} \in \mathbb{Z}^{n\times 1}$, find $g = \gcd(\boldsymbol{A})$, the gcd of all entries in $\boldsymbol{A}$. [8, Corollary 6.5] gives an $O(n\,\mathrm{B}(\log ||\boldsymbol{A}||))$ bit operations algorithm that solves the vector extended gcd problem, producing a $\boldsymbol{C} \in \mathbb{Z}^{1\times n}$ that satisfies $||\boldsymbol{C}|| \leq ||\boldsymbol{A}||$.

In this paper, we consider the special case $m = n - 1 = r$, that is, the input is a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$. To the best of our knowledge, the fastest previous algorithm to compute a basis $\boldsymbol{B}$ with $\log ||\boldsymbol{B}|| \in O(\log n + \log ||\boldsymbol{A}||)$ is that of Li and Nguyen [6, Theorem 3.8], which solves the problem in the general case, that is, for n, m and r arbitrary. Applied to the special case $m = n - 1 = r$ that we consider here, their algorithm has cost bounded by $O(n^\omega\,\mathrm{B}(nd))$ bit operations, where $d = \log n + \log ||\boldsymbol{A}||$ and ω is the exponent of matrix multiplication. In this paper, we give an algorithm that solves the problem in $O(n^\omega\,\mathrm{B}(d)(\log n)^2)$ bit operations. We remark that this cost estimate matches that of the Smith form algorithm of Birmpilis et al. [2] and thus our technique can be used for the application mentioned earlier, at least for the special case $m = n - 1 = r$.

Instead of producing $\boldsymbol{B}$ directly, we first compute a matrix $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ and then set $\boldsymbol{B} := \boldsymbol{C}\boldsymbol{A}$. We show that $\log ||\boldsymbol{C}|| \leq 4\log n + 2\log ||\boldsymbol{A}||$ and that $\boldsymbol{C}$ will have at most $n(1+\log_2 n)$ nonzero entries, independent of $||\boldsymbol{A}||$. To the best of our knowledge, we are not aware

that the existence of such a C with only $O(n \log n)$ nonzero entries was previously known.

The sparsity of C can be useful in some situations. Suppose, in addition to the full column rank input matrix $A \in \mathbb{Z}^{n\times(n-1)}$, we have additional linearly dependent columns $\bar{A} \in \mathbb{Z}^{n\times m}$. If $C \in \mathbb{Z}^{(n-1)\times n}$ is such that CA is a basis for A, then a basis for the rank $n-1$ augmented matrix $\left[\begin{array}{c|c} A & \bar{A} \end{array}\right] \in \mathbb{Z}^{n\times((n-1)+m)}$ is given by $C\left[\begin{array}{c|c} A & \bar{A} \end{array}\right]$. The additional cost to compute $C\bar{A}$ is only $O(nm\,\mathsf{M}(d)\log n)$ bit operations, where M is a cost function for integer multiplication, and $d = \log n + \log||\bar{A}||$.

We now give an outline of our approach to compute C. Our idea is to find a vector $v \in \mathbb{Z}^{n\times 1}$ such that the augmented matrix

$$\left[\begin{array}{c|c} A & v \end{array}\right] \in \mathbb{Z}^{n\times n} \tag{1}$$

is nonsingular. Then we compute $C \in \mathbb{Z}^{(n-1)\times n}$ to be a left kernel basis of v that satisfies $||C|| \le ||v||^2$. Of course, the augmentation vector v must be special in order for this to work. We show that if $\mathcal{L}(\left[\begin{array}{c|c} A & v \end{array}\right])$ contains the last row of I_n, then a left kernel basis $C \in \mathbb{Z}^{(n-1)\times n}$ of v has the property that CA is a basis for A. Our algorithm has three main computational steps:

(1) Compute the left kernel basis $w \in \mathbb{Z}^{1\times n}$ of A.
(2) Compute an augmentation vector $v \in \mathbb{Z}^{n\times 1}$ from A and w.
(3) Compute a left kernel basis $C \in \mathbb{Z}^{(n-1)\times n}$ of v.

The rest of this paper is organised as follows. Section 2 gives basic mathematical background. Section 3 establishes, via a more general result, the sufficient condition on the augmentation vector v as described above. Section 4 adapts from the literature some computational tools related to deterministic system solving. Section 5 uses these computational tools to obtain deterministic algorithms for the kernel basis problem (step 1). Section 6 gives the algorithm to produce the augmentation vector v (step 2). Section 7 gives the algorithm to compute a left kernel bases of v (step 3) and also states the main result of the paper.

Cost model

Following von zur Gathen and Gerhard [3, Section 8.3], cost estimates are given using a function $\mathsf{M}(d)$ that bounds the number of bit operations required to multiply two integers bounded in magnitude by 2^d. We use $\mathsf{B}(d)$ to bound the cost of integer gcd-related computations such as the extended euclidean algorithm. We can always take $\mathsf{B}(d) = O(\mathsf{M}(d)\log d)$. If $\mathsf{M}(d) \in \Omega(d^{1+\epsilon})$ for some $\epsilon > 0$, then $\mathsf{B}(d) \in O(\mathsf{M}(d))$.

As usual, we assume that M is superlinear and subquadratic. We also assume that $\mathsf{M}(ab) \in O(\mathsf{M}(a)\mathsf{M}(b))$ for $a, b \ge 1$. We assume that $\omega > 2$, and to simplify cost estimates, we make the common assumption that $\mathsf{M}(d) \in O(d^{\omega-1})$. The assumptions stated in this paragraph apply also to B.

2 MATHEMATICAL BACKGROUND

An important notion for us is that of a primitive integer matrix.

Definition 1 (Primitive matrix). *A matrix $A \in \mathbb{Z}^{n\times m}$ is* primitive *if one of the following conditions hold.*

(i) $(n > m)$ A has full column rank m and I_m is a basis for A.
(ii) $(n < m)$ A has full row rank n and I_n is a basis for Transpose(A).
(iii) $(n = m)$ A is unimodular (that is, $\det A = \pm 1$*).*

Definition 2 (Left kernel basis of an integer matrix). *A* left kernel basis *of an $A \in \mathbb{Z}^{n\times m}$ with rank of r is a matrix $K \in \mathbb{Z}^{(n-r)\times n}$ such that*

(i) K has rank $n-r$,
(ii) KA is the $(n-r)\times m$ zero matrix, and
(iii) K is primitive.

We remark that a matrix K that satisfies only properties (i) and (ii) of Definition 2 is a nullspace basis of A over $\mathbb{Q}$: property (ii) ensures that the set of all $\mathbb{Q}$-linear combinations of the rows of K generates $\{v \in \mathbb{Q}^{1\times n} \mid vA = 0_{1\times m}\}$. The additional property (iii) in Definition 2 ensures that K satisfies $\mathcal{L}(K) = \{v \in \mathbb{Z}^{1\times n} \mid vA = 0_{1\times m}\}$.

Lemma 3. *Let $A \in \mathbb{Z}^{n\times m}$ have full column rank. If $B \in \mathbb{Z}^{m\times m}$ is a basis for A, then AB^{-1} is integral and primitive.*

Proof. Since B is a basis for A, B is nonsingular and we have

$$UA = \left[\begin{array}{c} B \\ \hline \\ \end{array}\right] \tag{2}$$

for some unimodular matrix $U \in \mathbb{Z}^{n\times n}$. Postmultiplying both sides of (2) by B^{-1} gives

$$UAB^{-1} = \left[\begin{array}{c} I_m \\ \hline \\ \end{array}\right]. \tag{3}$$

Premultiplying both sides of equation (3) by U^{-1} gives

$$AB^{-1} = U^{-1}\left[\begin{array}{c} I_m \\ \hline \\ \end{array}\right]. \tag{4}$$

Since U is unimodular, U^{-1} is integral. The right side of equation (4) is integral. Hence, AB^{-1} is integral. It follows from equation (3) that AB^{-1} is primitive. □

Lemma 4. *Let $A \in \mathbb{Z}^{n\times m}$ of rank r be given. If $C \in \mathbb{Z}^{r\times n}$ is such that CA is a basis for A, and $K \in \mathbb{Z}^{(n-r)\times n}$ is a left kernel basis of A, then*

$$\left[\begin{array}{c} C \\ \hline K \end{array}\right] \in \mathbb{Z}^{n\times n} \tag{5}$$

is unimodular.

Proof. Since K is primitive, it follows from Definition 1.(ii) that there exists a unimodular matrix U such that $KU = \left[\begin{array}{c|c} & I_{n-r} \end{array}\right]$. Let

$$\left[\begin{array}{c} C \\ K \end{array}\right] U = \left[\begin{array}{c|c} V & * \\ \hline & I_{n-r} \end{array}\right] \tag{6}$$

and

$$U^{-1}A = \left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right], \tag{7}$$

where $A_1 \in \mathbb{Z}^{r\times m}$. Multiplying the right hand side of (6) by that of (7) gives

$$\left[\begin{array}{c|c} V & * \\ \hline & I_{n-r} \end{array}\right]\left[\begin{array}{c} A_1 \\ \hline A_2 \end{array}\right] = \left[\begin{array}{c} CA \\ \hline \\ \end{array}\right]. \tag{8}$$

We conclude from (8) that A_2 is the zero matrix and then from (7) that A_1 is a basis for A. Since CA is also a basis for A, the matrix V must be unimodular. The result now follows from (6). □

3 LATTICE GENERATOR VIA KERNEL BASIS

Suppose $A \in \mathbb{Z}^{n\times(n-1)}$ and $v \in \mathbb{Z}^{n\times 1}$ are such that $\left[\begin{array}{c|c} A & v \end{array}\right]$ is nonsingular. In this section, we establish a sufficient condition on v to ensure that a left kernel basis $C \in \mathbb{Z}^{(n-1)\times n}$ of v has the property that CA is a basis for A.

We rely on the following theorem that applies only to those nonsingular matrices $A \in \mathbb{Z}^{n\times n}$ for which there exists a unimodular decoupling matrix $U \in \mathbb{Z}^{n\times n}$ such that

$$UA = \left[\begin{array}{c|c} *_1 & \\ \hline & *_2 \end{array}\right].$$

Note that for such an A, we have $\mathcal{L}(A) = \left\{\left[\begin{array}{c|c} \mathcal{L}(*_1) & \mathcal{L}(*_2) \end{array}\right]\right\}$.

Theorem 5. *Let $A = \left[\begin{array}{c|c} A_1 & A_2 \end{array}\right] \in \mathbb{Z}^{n\times n}$ be nonsingular, where $A_1 \in \mathbb{Z}^{n\times m_1}$ and $A_2 \in \mathbb{Z}^{n\times m_2}$. For any $C_1 \in \mathbb{Z}^{m_1\times n}$ and $C_2 \in \mathbb{Z}^{m_2\times n}$, we have*

(a) $\left[\begin{array}{c} C_1 \\ \hline C_2 \end{array}\right]\left[\begin{array}{c|c} A_1 & A_2 \end{array}\right] = \left[\begin{array}{c|c} C_1A_1 & \\ \hline & C_2A_2 \end{array}\right] \in \mathbb{Z}^{n\times n}$, with
(b) C_1A_1 a basis for A_1, and
(c) C_2A_2 a basis for A_2

if and only if

(i) $\left[\begin{array}{c} C_1 \\ \hline C_2 \end{array}\right] \in \mathbb{Z}^{n\times n}$ is unimodular,
(ii) C_1 is a kernel basis for A_2, and
(iii) C_2 is a kernel basis for A_1.

Proof. **(Only if:)** Assume that (a), (b) and (c) hold. Since $A = \left[\begin{array}{c|c} A_1 & A_2 \end{array}\right]$ is nonsingular, A_i has full column rank ($i = 1, 2$). From (b) and (c), we have that $C_iA_i \in \mathbb{Z}^{m_i\times m_i}$ also has full column rank ($i = 1, 2$). It follows that C_iA_i is nonsingular ($i = 1, 2$).

Postmultiplying both sides of the equation in (a) by

$$\left[\begin{array}{c|c} (C_1A_1)^{-1} & \\ \hline & (C_2A_2)^{-1} \end{array}\right]$$

gives

$$\left[\begin{array}{c} C_1 \\ \hline C_2 \end{array}\right]\left[\begin{array}{c|c} A_1\,(C_1A_1)^{-1} & A_2\,(C_2A_2)^{-1} \end{array}\right] = \left[\begin{array}{c|c} I_{m_1} & \\ \hline & I_{m_2} \end{array}\right]. \quad (9)$$

Since (C_iA_i) is a basis for A_i ($i = 1, 2$), it follows from Lemma 3 that

$$U := \left[\begin{array}{c|c} A_1\,(C_1A_1)^{-1} & A_2\,(C_2A_2)^{-1} \end{array}\right]$$

is an integer matrix. Therefore,

$$\det\left(\left[\begin{array}{c} C_1 \\ \hline C_2 \end{array}\right]\right) = \pm 1$$

and property (i) holds. It remains to show that (ii) and (iii) hold. Since U is an integer matrix, it follows from (9) that C_1 satisfies the properties required by Definition 2 to be a left kernel of A_2. In particular, C_1 is primitive and $C_1A_2 = 0_{m_1\times m_2}$. Similarly, C_2 is a left kernel of A_1.

(If:) Assume that (i), (ii) and (iii) hold. It follows from (ii) and (iii) that (a) holds. It now follows from property (i) that

$$\mathcal{L}\left(\left[\begin{array}{c|c} A_1 & A_2 \end{array}\right]\right) = \mathcal{L}\left(\left[\begin{array}{c|c} C_1A_1 & \\ \hline & C_2A_2 \end{array}\right]\right).$$

Properties (b) and (c) now follow by noting that A_i and C_iA_i have the same column dimension ($i = 1, 2$). □

The following corollary identifies a particular class of matrices for which Theorem 5 applies.

Corollary 6. *Let $A = \left[\begin{array}{c|c} A_1 & A_2 \end{array}\right] \in \mathbb{Z}^{n\times n}$ be nonsingular, where $A_1 \in \mathbb{Z}^{n\times m_1}$ and $A_2 \in \mathbb{Z}^{n\times m_2}$. If $\mathcal{L}(A)$ contains the last m_2 rows of I_n, then any left kernel basis $C \in \mathbb{Z}^{m_1\times n}$ of A_2 has the property that CA_1 is a basis for A_1.*

Proof. Since $\mathcal{L}(A)$ contains the last m_2 rows of I_n, there exists a unimodular matrix $U \in \mathbb{Z}^{n\times n}$ such that

$$UA = \left[\begin{array}{c|c} * & \\ \hline & I_{m_2} \end{array}\right].$$

Decompose U as

$$U = \left[\begin{array}{c} C_1 \\ \hline C_2 \end{array}\right],$$

where $C_1 \in \mathbb{Z}^{m_1\times n}$ and $C_2 \in \mathbb{Z}^{m_2\times n}$. Then the equation in property (a) of Theorem 5 holds, and from the unimodularity of U, it follows that C_1 is a left kernel basis for A_2, and C_2 is a left kernel basis for A_1. We are thus exactly in the situation of Theorem 5, and can conclude that C_1A_1 is a basis for A_1. Since any two left kernel bases of A_2 are left equivalent, we can replace C_1 with C in U and the matrix remains unimodular. □

4 COMPUTATIONAL TOOLS

The main computational tool we need is an algorithm for nonsingular linear system solving. A deterministic algorithm has recently been given by Birmpilis et al. [1, Section 11], but was analyzed under a more restrictive cost model than we are using in this paper. We restate the result here.

Theorem 7 (System solving). *There exists an algorithm that takes as input a nonsingular matrix $A \in \mathbb{Z}^{n\times n}$ and a vector $b \in \mathbb{Z}^{n\times 1}$, and returns as output $A^{-1}b \in \mathbb{Q}^{n\times 1}$, together with the minimal $e \in \mathbb{Z}_{>0}$ such that $eA^{-1}b$ is integral. Let $d = \log n + \log||A||$. If $\log||b|| \in O(nd)$, the running time of the algorithm is $O(n^\omega\,\mathrm{B}(d)(\log n)^2)$ bit operations.*

Proof. [1, Section 11] gives an algorithm that computes an integer vector $x \in \mathbb{Z}^{n\times 1}$ together with an $f \in \mathbb{Z}_{\geq 0}$ such that $2^fA^{-1}b$ can be recovered from x using rational number reconstruction. The cost bound of [1, Theorem 22] is

$$O(n^\omega\,\mathrm{M}(d)(\log n + \mathrm{loglog}\,||A||)(\log n)). \quad (10)$$

The cost model used in [1] does not make the assumptions that $\omega > 2$ and $\mathrm{M}(d) \in O(d^{\omega-1})$. If we do make these assumptions, and we replace the subroutine in [1, Section 6] with the integer analogue of the algorithm supporting [4, Theorem 7], the loglog $||A||$ term in (10) can be avoided, giving the cost bound $O(n^\omega\,\mathrm{M}(d)(\log n)^2)$ to recover x.

To recover $A^{-1}b$ from x requires rational number reconstruction. By [3, Section 5.10], the cost of computing e and $eA^{-1}b$ is

$O(n\,\mathrm{B}(nd))$ bit operations; this simplifies to $O(n^\omega\,\mathrm{B}(d))$ using the assumptions $\mathrm{B}(nd) \in O(\mathrm{B}(n)\mathrm{B}(d))$ and $\mathrm{B}(n) \in O(n^{\omega-1}\,\mathrm{B}(d))$. □

The algorithm supporting Theorem 7 was based on computing a 2-massager for the input matrix. Although originally defined for nonsingular matrices, the notion of a 2-massager extends directly to matrices of full column rank.

Definition 8 (2-Smith form). *Let $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ have full column rank m. The* 2-Smith form *of $\boldsymbol{A}$ is the matrix* $\mathrm{diag}(2^{e_1}, 2^{e_2}, ..., 2^{e_m})$ *with 2^{e_i} the largest power of two which divides the invariant factor i of the Smith form of $\boldsymbol{A}$ over $\mathbb{Z}$, $1 \le i \le m$.*

Definition 9. *Let $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ have full column rank m. A* 2-massager *for $\boldsymbol{A}$ is a triple of matrices $(\boldsymbol{P}, \boldsymbol{S}, \boldsymbol{M})$ from $\mathbb{Z}^{m\times m}$ such that,*

- *$\boldsymbol{P}$ is a permutation,*
- *$\boldsymbol{S} = \mathrm{diag}(s_1, \ldots, s_m)$ is the 2-Smith form of $\boldsymbol{A}$,*
- *$\boldsymbol{M}$ is unit upper triangular, and*
- *$\boldsymbol{APMS}^{-1}$ is integral with $\mathrm{Rem}(\boldsymbol{APMS}^{-1}, 2) \in \mathbb{Z}/(2)^{n\times m}$ having full column rank over $\mathbb{Z}/(2)$.*

$(\boldsymbol{P}, \boldsymbol{S}, \boldsymbol{M})$ is a reduced 2-massager *if entries in column i of $\boldsymbol{M}$ are from $[0, \ldots s_i - 1]$, $1 \le i \le n$.*

Example 10. *A 2-massager for the the matrix*

$$\boldsymbol{A} := \begin{bmatrix} 92 & 27 & 99 \\ -124 & 32 & 116 \\ 556 & -42 & -150 \\ 176 & 249 & 77 \\ -8 & 195 & -121 \end{bmatrix} \in \mathbb{Z}^{5\times 3}$$

is given by

$$\boldsymbol{P}, \boldsymbol{S}, \boldsymbol{M} := \begin{bmatrix} & & 1 \\ 1 & & \\ & 1 & \end{bmatrix}, \begin{bmatrix} 1 & & \\ & 4 & \\ & & 16 \end{bmatrix}, \begin{bmatrix} 1 & 3 & 5 \\ & 1 & 15 \\ & & 1 \end{bmatrix}.$$

The massaged matrix

$$\boldsymbol{B} := \boldsymbol{APMS}^{-1} = \begin{bmatrix} 27 & 45 & 107 \\ 32 & 53 & 111 \\ -42 & -69 & -119 \\ 249 & 206 & 161 \\ 195 & 116 & -53 \end{bmatrix}$$

has full column rank over $\mathbb{Z}/(2)$.

Birmpilis et al. [1, Section 10] give an algorithm to compute a 2-massager of a nonsingular matrix. The algorithm extends naturally to an $n \times m$ matrix of full column rank m.

Theorem 11 (2-massager computation). *There exists an algorithm that takes as input a full column rank $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, and returns as output a reduced 2-massager for $\boldsymbol{A}$ together with the massaged matrix $\boldsymbol{B} := \boldsymbol{APMS}^{-1}$. The cost of the algorithm is $O(nm^{\omega-1}\,\mathsf{M}(d)(\log m)^2)$ bit operations, where $d = \log m + \log ||\boldsymbol{A}||$.*

Proof. Similar to the proof of Theorem 7, if we replace the subroutine in [1, Section 6] with the integer analogue of the algorithm supporting [4, Theorem 7], then [1, Theorem 21] establishes the theorem in the case $n = m$, that is, for a square nonsingular $m \times m$ input matrix, a 2-massager can be computed in

$$O(m^\omega\,\mathsf{M}(d)(\log m)^2)$$

bit operations. The only required modifications now to handle case of a full column rank rectangular $n \times m$ input matrix is to incorporate blocking into steps 1–4 of the algorithm supporting [1, Theorem 19].

Step 1 computes an LQUP decomposition of an $m \times m$ matrix over $\mathbb{Z}/(2)$ at a cost of $O(n^\omega)$ bit operations. In the rectangular case this costs $O(nm^{\omega-1})$ bit operations.

Step 2 computes a single nonsingular system solution of dimension m. In the rectangular case, step 2 can be accomplished by computing $\lceil n/m \rceil$ such linear systems, thus replacing the m^ω term in the cost estimate for the square case by $nm^{\omega-1}$.

Steps 3 and 4 compute a triangular Smith form of a square matrix over ring. In the rectangular case, these steps are accomplished using the integer analogue of [4, Theorem 7] followed up by the triangularization of a matrix that, compared to the square case, has row dimension now bounded by n. The cost increases similarly as for step 2. □

One additional algorithm that we need is a fast solution to the vector extended gcd problem [8, Corollary 6.5].

Algorithm 1: `VectorGcdex(`$\boldsymbol{w}$`)`

Input : $\boldsymbol{w} \in \mathbb{Z}^{n\times 1}$

Output : $(\boldsymbol{e}, g) \in (\mathbb{Z}^{1\times n}, \mathbb{Z})$, satisfying

- $\boldsymbol{e}\boldsymbol{w} = g$, with $g = \gcd(\boldsymbol{w})$,
- $||\boldsymbol{e}|| \le ||\boldsymbol{w}||$, and
- the number of nonzero entries in $\boldsymbol{e}$ is bounded by $1 + \log_2 ||\boldsymbol{w}||$.

Runtime : $O(n\,\mathrm{B}(\log ||\boldsymbol{w}||))$ bit operations

5 KERNEL BASIS OF NULLITY ONE

We give an algorithm to compute a left kernel basis $\boldsymbol{w} \in \mathbb{Z}^{1\times n}$ of a full column rank input matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$. In this special case, where the nullity of $\boldsymbol{A}$ is one, the kernel basis $\boldsymbol{w}$ is unique (up to sign) and can be computed using nonsingular linear system solving. If $\boldsymbol{Q} \in \mathbb{Z}^{n\times n}$ is a permutation such that

$$\boldsymbol{QA} = \left[\frac{\bar{\boldsymbol{A}}}{\boldsymbol{a}}\right] \in \mathbb{Z}^{n\times(n-1)} \tag{11}$$

with $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(n-1)\times(n-1)}$ nonsingular, then a left nullspace of $\boldsymbol{QA}$ over $\mathbb{Q}$ is

$$\left[\ \boldsymbol{a}\bar{\boldsymbol{A}}^{-1} \ \middle|\ -1\ \right] \in \mathbb{Q}^{1\times n}.$$

If $e \in \mathbb{Z}_{>0}$ is minimal such that $e\boldsymbol{a}\bar{\boldsymbol{A}}^{-1}$ is integral, then

$$\boldsymbol{w} = \boldsymbol{Q}\left[\ e\boldsymbol{a}\bar{\boldsymbol{A}}^{-1} \ \middle|\ -e\ \right] \in \mathbb{Z}^{1\times n} \tag{12}$$

is primitive and is thus a kernel basis of $\boldsymbol{A}$ (Definition 2).

Theorem 12 (Kernel of nullity one). *A left kernel basis of $\boldsymbol{w} \in \mathbb{Z}^{1\times n}$ of a full column rank $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ can be computed in $O(n^\omega\,\mathrm{B}(d)(\log n)^2)$ bit operations, where $d = \log n + \log ||\boldsymbol{A}||$.*

Proof. To find a suitable $\boldsymbol{Q}$, compute a 2-massager $(\boldsymbol{P}, \boldsymbol{M}, \boldsymbol{S})$ of $\boldsymbol{A}$ together with the massaged matrix $\boldsymbol{B} = \boldsymbol{APMS}^{-1}$. The massaged matrix $\boldsymbol{B}$ has full column rank modulo 2. Working modulo 2, use

the LSP decomposition of Ibarra et al. [5, Theorem 3.2] to compute the row rank profile $[i_1, i_2, \ldots, i_{n-1}]$ of $\boldsymbol{B}$ over $\mathbb{Z}/(2)$. Since $\boldsymbol{PMS}^{-1}$ is nonsingular, the submatrix of $\boldsymbol{A}$ comprised of rows $i_1, i_2, \ldots, i_{n-1}$ is also nonsingular. Define $\boldsymbol{Q}$ accordingly.

Compute the minimal $e \in \mathbb{Z}_{>0}$ such that $e\boldsymbol{a}\bar{\boldsymbol{A}}^{-1}$ is integral, and compute $e\boldsymbol{a}\bar{\boldsymbol{A}}^{-1}$ itself. Return $\boldsymbol{w}$ as in (12). The claimed cost bound follows from Theorems 11 (2-massager computation) and 7 (System solving). □

The following corollary follows from Hadamard's inequality [3, Theorem 16.6] and Cramer's rule [3, Theorem 25.6].

Corollary 13. *The kernel basis produced by Theorem 12 satisfies* $\log ||\boldsymbol{w}|| \in O(nd)$.

6 AUGMENTATION VECTOR

Storjohann [9, Section 13] describes a Las Vegas randomized algorithm `DetReduction` that takes as input a nonsingular integer input matrix, and replaces the last column to obtain a new matrix whose lattice contains the last row of $\boldsymbol{I}$. Algorithm `DetReduction` exploits the fact that the input matrix is nonsingular. Here, we adapt the approach to obtain the following result. In the proof, we focus mainly on showing how to avoid the requirement that the input matrix be nonsingular, and how to avoid randomization by using the subroutines developed in previous sections.

Theorem 14 (Augmentation vector). *There exists an algorithm that takes as input a full column rank integer matrix* $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, *and returns as output an integer vector* $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ *such that*

(1) $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ *contains the last row of* $\boldsymbol{I}_n$, *and*
(2) $||\boldsymbol{v}|| \le n^2||\boldsymbol{A}||$.

The cost of the algorithm is $O(n^\omega\, \mathrm{B}(d)(\log n)^2)$ *bit operations, where* $d = \log n + \log ||\boldsymbol{A}||$.

Proof. Compute a kernel basis $\boldsymbol{w} \in \mathbb{Z}^{1\times n}$ of $\boldsymbol{A}$. Let $\boldsymbol{P} \in \mathbb{Z}^{n\times n}$ be a permutation matrix such that $\boldsymbol{wP}$ has last entry of maximal magnitude. Then

$$(\boldsymbol{wP})(\boldsymbol{P}^{-1}\boldsymbol{A}) = \left[\begin{array}{c|c} \bar{\boldsymbol{w}} & w_n \end{array}\right] \left[\begin{array}{c} \bar{\boldsymbol{A}} \\ \hline \boldsymbol{a} \end{array}\right] = \boldsymbol{0}_{1\times(n-1)}$$

where $|w_n| = ||\boldsymbol{w}||$ and $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(n-1)\times(n-1)}$ is nonsingular since $w_n \neq 0$. Going forward, we will assume, without loss of generality, that $\boldsymbol{P} = \boldsymbol{I}_n$.

Use Algorithm 1 (`VectorGcdex`) to compute a vector $\boldsymbol{b} \in \mathbb{Z}^{n\times 1}$ such that

$$\boldsymbol{w}\,\boldsymbol{b} = \left[\begin{array}{c|c} \bar{\boldsymbol{w}} & w_n \end{array}\right] \left[\begin{array}{c} \bar{\boldsymbol{b}} \\ \hline b_n \end{array}\right] = 1.$$

Then

$$\left[\begin{array}{c|c} \bar{\boldsymbol{w}} & w_n \end{array}\right] \left[\begin{array}{c|c} \bar{\boldsymbol{A}} & \bar{\boldsymbol{b}} \\ \hline \boldsymbol{a} & b_n \end{array}\right] = \left[\begin{array}{c|c} & 1 \end{array}\right]. \quad (13)$$

It follows from (13) that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{b} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$, and thus $\boldsymbol{b}$ is candidate for our solution vector. However, the entries of $\boldsymbol{b}$ can be too large.

We now show how to adjust $\boldsymbol{b}$ to obtain a reduced solution $\boldsymbol{v}$. Postmultiplying both sides of equation (13) by a unimodular matrix

$$\left[\begin{array}{c|c} \boldsymbol{I}_{n-1} & -\bar{\boldsymbol{y}} \\ \hline & 1 \end{array}\right] \in \mathbb{Z}^{n\times n}$$

gives

$$\left[\begin{array}{c|c} \bar{\boldsymbol{w}} & w_n \end{array}\right] \left[\begin{array}{c|c} \bar{\boldsymbol{A}} & \bar{\boldsymbol{b}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{y}} \\ \hline \boldsymbol{a} & b_n - \boldsymbol{a}\bar{\boldsymbol{y}} \end{array}\right] = \left[\begin{array}{c|c} & 1 \end{array}\right]. \quad (14)$$

Let $\boldsymbol{y} \in \mathbb{Q}^{(n-1)\times 1}$ be the solution to the linear system $\bar{\boldsymbol{A}}\boldsymbol{y} = \bar{\boldsymbol{b}}$. Let $\bar{\boldsymbol{y}} \in \mathbb{Z}^{(n-1)\times n}$ be the integral part of $\boldsymbol{y}$, that is, $\bar{\boldsymbol{y}}$ is the unique integer vector such that $||\boldsymbol{y} - \bar{\boldsymbol{y}}|| < 1$. Then $||\bar{\boldsymbol{b}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{y}}|| \le n||\boldsymbol{A}||$. We choose our solution vector to be $\boldsymbol{v} := \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}$. It remains to show that the last entry $v_n = b_n - \boldsymbol{a}\bar{\boldsymbol{y}}$ of $\boldsymbol{v}$ has magnitude bounded by $n^2||\boldsymbol{A}||$. From equation (14), we have $\bar{\boldsymbol{w}}\,\bar{\boldsymbol{v}} + w_n v_n = 1$. It follows that

$$|v_n| = \left\| \frac{\bar{\boldsymbol{w}}}{w_n}\bar{\boldsymbol{v}} - \frac{1}{w_n} \right\| \le \left\| \frac{\bar{\boldsymbol{w}}}{w_n}\bar{\boldsymbol{v}} \right\| + \left| \frac{1}{w_n} \right|.$$

Since $\bar{\boldsymbol{w}}$ has column dimension $n-1$, $||\bar{\boldsymbol{w}}|| \le |w_n| = ||\boldsymbol{w}||$, and $||\bar{\boldsymbol{v}}|| \le n||\boldsymbol{A}||$, we have

$$|v_n| \le (n-1)||\bar{\boldsymbol{v}}|| + 1 \le (n-1)n||\boldsymbol{A}|| + 1 \le n^2||\boldsymbol{A}||.$$

Corollary 13 gives the bound $\log ||\boldsymbol{w}|| \in O(nd)$. From the specification of Algorithm 1 (VectorGcdex), we have $||\boldsymbol{b}|| \le ||\boldsymbol{w}||$, so $\log ||\boldsymbol{b}|| \in O(nd)$ also. The claimed running time bound now follows from Theorem 12 (Kernel of nullity one), the running time of Algorithm 1 (VectorGcdex), and Theorem 7 (System solving). □

Example 15. *The input matrix*

$$\boldsymbol{A} = \begin{bmatrix} 231 & 303 & -118 & 16 \\ 344 & 202 & -389 & 163 \\ 185 & 190 & -80 & 136 \\ 263 & 189 & 196 & 66 \\ 259 & 157 & 131 & 136 \end{bmatrix} \in \mathbb{Z}^{5\times 4}$$

has left kernel

$$\boldsymbol{w} = \begin{bmatrix} -3330033 & 825718 & 4373666 & 7018431 & -8377548 \end{bmatrix}.$$

`VectorGcdex` *computes a* $\boldsymbol{b} \in \mathbb{Z}^{5\times 1}$ *such that*

$$\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{b} \end{array}\right] = \left[\begin{array}{c|c} \bar{\boldsymbol{A}} & \bar{\boldsymbol{b}} \\ \hline \boldsymbol{a} & b_5 \end{array}\right] = \left[\begin{array}{cccc|c} 231 & 303 & -118 & 16 & -2022969 \\ 344 & 202 & -389 & 163 & 0 \\ 185 & 190 & -80 & 136 & 2 \\ 263 & 189 & 196 & 66 & 0 \\ \hline 259 & 157 & 131 & 136 & 804121 \end{array}\right]$$

with $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{b} \end{array}\right])$ *containing the last row of* $\boldsymbol{I}_5$. *Decompose* $\boldsymbol{y} = \bar{\boldsymbol{A}}^{-1}\bar{\boldsymbol{b}} = \bar{\boldsymbol{y}} + \boldsymbol{r}$ *where* $\bar{\boldsymbol{y}} \in \mathbb{Z}^{4\times 1}$ *and* $\boldsymbol{r} \in \mathbb{Q}^{4\times 1}$ *with* $||\boldsymbol{r}|| < 1$:

$$\boldsymbol{y} = \begin{bmatrix} \frac{219432205277}{94247415} \\ -\frac{10934282147281}{1319463810} \\ \frac{896384538211}{527785524} \\ \frac{709438547431}{75397932} \end{bmatrix} = \bar{\boldsymbol{y}} + \boldsymbol{r} = \begin{bmatrix} 2328 \\ -8286 \\ 1698 \\ 9409 \end{bmatrix} + \begin{bmatrix} \frac{24223157}{94247415} \\ -\frac{1205017621}{1319463810} \\ \frac{204718459}{527785524} \\ \frac{19405243}{75397932} \end{bmatrix}.$$

Our augmentation vector is given by

$$\boldsymbol{v} = \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}} = \begin{bmatrix} -259 \\ -205 \\ -122 \\ -12 \\ 9 \end{bmatrix}.$$

7 GENERATING A LATTICE BASIS

We begin by giving an algorithm to compute a left kernel basis of a vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$. The algorithm we give is closely based on a more general algorithm [8, Chapter 6.2] that computes a left kernel basis for an $n \times m$ integer matrix. The variation we give here for the special case $m = 1$ is simpler and allows us to obtain a tighter explicit bound on the number of nonzero entries in the kernel.

Algorithm 2: `SparseKernelBasis`$(\boldsymbol{v}, n)$

Input : $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ with all entries nonzero

Output : $\boldsymbol{K} \in \mathbb{Z}^{(n-1)\times n}$, a left kernel basis of $\boldsymbol{v}$ satisfying
- $||\boldsymbol{K}|| \le ||\boldsymbol{v}||^2$, and
- the number of nonzero entries in $\boldsymbol{K}$ is bounded by $n(1 + \log_2 n)$.

Runtime : $O(n\,\mathrm{B}(\log ||\boldsymbol{v}||) \log n)$ bit operations

if $n = 1$ **then**
  return the 0×1 matrix
$n_1 := \lfloor n/2 \rfloor$
$n_2 := n - n_1$
Decompose $\boldsymbol{v} = \left[\begin{array}{c} \boldsymbol{v}_1 \\ \hline \boldsymbol{v}_2 \end{array}\right]$ where $\boldsymbol{v}_1 \in \mathbb{Z}^{n_1\times 1}$ and $\boldsymbol{v}_2 \in \mathbb{Z}^{n_2\times 1}$
$\boldsymbol{K}_1 :=$ `SparseKernelBasis`$(\boldsymbol{v}_1, n_1)$
$\boldsymbol{K}_2 :=$ `SparseKernelBasis`$(\boldsymbol{v}_2, n_2)$
$\boldsymbol{e}_1, g_1 :=$ `VectorGcdex`$(\boldsymbol{v}_1, n_1)$
$\boldsymbol{e}_2, g_2 :=$ `VectorGcdex`$(\boldsymbol{v}_2, n_2)$
$g := \gcd(g_1, g_2)$
return $\left[\begin{array}{c|c} -(g_2/g)\,\boldsymbol{e}_1 & (g_1/g)\,\boldsymbol{e}_2 \\ \hline \boldsymbol{K}_1 & \\ \hline & \boldsymbol{K}_2 \end{array}\right]$

We will prove correctness of Algorithm `SparseKernelBasis` using three lemmas. The first lemma shows that the output is indeed a kernel basis. The second lemma establishes that the output satisfies the two stated conditions. The third lemma gives an analysis of the running time.

Lemma 16. `SparseKernelBasis` *outputs a kernel basis for* $\boldsymbol{v}$.

Proof. The algorithm uses a divide and conquer approach. The output of the base case ($n = 1$) is correct since $\boldsymbol{v} \in \mathbb{Z}^{1\times 1}$ is nonzero by assumption.

For the recursive case ($n > 1$), we have subproblems of size n_1 and n_2, with $n = n_1 + n_2$. By Lemma 4, the matrix

$$\boldsymbol{U}_i = \left[\begin{array}{c} \boldsymbol{e}_i \\ \hline \\ \boldsymbol{K}_i \\ \\ \end{array}\right] \in \mathbb{Z}^{n_i \times n_i}$$

is unimodular ($i = 1, 2$). The matrix $\boldsymbol{U} := \mathrm{diag}(\boldsymbol{U}_1, \boldsymbol{U}_2) \in \mathbb{Z}^{n\times n}$ is thus unimodular, and by construction we have

$$\overset{\boldsymbol{U}}{\left[\begin{array}{c|c} \boldsymbol{U}_1 & \\ \hline & \boldsymbol{U}_2 \end{array}\right]} \overset{\boldsymbol{v}}{\left[\begin{array}{c} \boldsymbol{v}_1 \\ \hline \boldsymbol{v}_2 \end{array}\right]} = \overset{\boldsymbol{c}}{\left[\begin{array}{c} g_1 \\ \hline g_2 \end{array}\right]} \in \mathbb{Z}^{n\times 1}, \tag{15}$$

where $\boldsymbol{c}$ contains only two nonzero entries, g_1 at index 1 and g_2 at index $n_1 + 1$.

Note that the vector $\begin{bmatrix} -g_2/g & g_1/g \end{bmatrix} \in \mathbb{Z}^{1\times 2}$ is primitive and thus can be extended to a unimodular matrix

$$\begin{bmatrix} *_1 & *_2 \\ -g_2/g & g_1/g \end{bmatrix} \in \mathbb{Z}^{2\times 2}$$

that satisfies

$$\begin{bmatrix} *_1 & *_2 \\ -g_2/g & g_1/g \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} g \\ \end{bmatrix}.$$

Let $\boldsymbol{E}$ be equal to $\boldsymbol{I}_n$ except with principal $(n_1{+}1)\times(n_1{+}1)$ submatrix equals to

$$\begin{bmatrix} *_1 & & & *_2 \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ -g_2/g & & & g_1/g \end{bmatrix} \in \mathbb{Z}^{(n_1+1)\times(n_1+1)}.$$

Then $\boldsymbol{E}$ is unimodular with

$$\overset{\boldsymbol{E}}{\begin{bmatrix} *_1 & & & *_2 & \\ & \ddots & & & \\ & & 1 & & \\ -g_2/g & & & g_1/g & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix}} \overset{\boldsymbol{c}}{\left[\begin{array}{c} g_1 \\ \hline g_2 \\ \\ \\ \end{array}\right]} = \overset{\bar{\boldsymbol{c}}}{\left[\begin{array}{c} g \\ \\ \\ \\ \\ \end{array}\right]} \tag{16}$$

By (15) and (16), $(\boldsymbol{EU})\boldsymbol{v} = \bar{\boldsymbol{c}}$. Since the last $n - 1$ entries of $\bar{\boldsymbol{c}}$ are zeros, and $\boldsymbol{EU}$ is unimodular, the last $n - 1$ rows of $\boldsymbol{EU}$ is a kernel basis of $\boldsymbol{v}$. The algorithm outputs the last $n - 1$ rows of $\boldsymbol{EU}$, except with row 1 and row $n_1 + 1$ interchanged. □

Lemma 17. *The output* $||\boldsymbol{K}||$ *of* `SparseKernelBasis` *satisfies:*
(i) $||\boldsymbol{K}|| \le ||\boldsymbol{v}||^2$, *and*
(ii) the number of nonzero entries in $\boldsymbol{K}$ *is bounded by* $n(1 + \log_2 n)$.

Proof. The algorithm splits the problem into two subproblems with input $\boldsymbol{v}_1 \in \mathbb{Z}^{\lceil n/2 \rceil \times 1}$, and $\boldsymbol{v}_2 \in \mathbb{Z}^{\lfloor n/2 \rfloor \times 1}$. The kernel basis $\boldsymbol{K}$ for $\boldsymbol{v}$ is comprised of the kernel basis $\boldsymbol{K}_i$ of $\boldsymbol{v}_i$ ($i = 1, 2$), and one extra row

$$\boldsymbol{e} := \left[\begin{array}{c|c} -(g_2/g)\,\boldsymbol{e}_1 & (g_1/g)\,\boldsymbol{e}_2 \end{array}\right] \in \mathbb{Z}^{1\times n},$$

which we focus on to establish properties (i) and (ii).

First consider property (i). Since $g_i = \gcd(\boldsymbol{v}_i)$, we have $|g_i/g| \leq ||\boldsymbol{v}||$ $(i = 1, 2)$. According to the output specification of Algorithm 1 (`VectorGcdex`), we have $||\boldsymbol{e}_i|| \leq ||\boldsymbol{v}_i|| \leq ||\boldsymbol{v}||$ $(i = 1, 2)$. It follows that $||\boldsymbol{e}|| \leq ||\boldsymbol{v}||^2$. Induction on n (base cases $n = 1, 2$) now shows that property (i) holds.

Now consider property (ii). Since the extra row $\boldsymbol{e}$ has at most n nonzero entries, an upper bound on the number of nonzero entries in $\boldsymbol{K}$ satisfies the recurrence

$$T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n, \quad T(1) = 0.$$

From Sloane and Inc. [7], we obtain that

$$T(n) = n(\lfloor \log_2 n \rfloor + 3) - 2^{\lceil \log_2 n \rceil + 1}.$$

Dropping the floor and ceiling and simplifying yields the claimed upper bound. □

Lemma 18. *The running time of* `SparseKernelBasis` *is bounded by* $O(n\,\mathrm{B}(\log ||\boldsymbol{v}||) \log n)$ *bit operations.*

Proof. The nonrecursive work is dominated by the calls to Algorithm 1 (`VectorGcdex`) which has cost $O(n\,\mathrm{B}(\log ||\boldsymbol{v}||))$. The running time of the algorithm thus satisfies the recurrence

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(n\,\mathrm{B}(\log ||\boldsymbol{v}||)),$$

which solves to the running time bound stated in the lemma. □

Theorem 19 (Sparse kernel basis). *There exists an algorithm that takes as input a nonzero vector* $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$*, and returns as output a left kernel basis* $\boldsymbol{K} \in \mathbb{Z}^{(n-1)\times n}$ *for* $\boldsymbol{v}$*. The cost of the algorithm is* $O(n\,\mathrm{B}(\log ||\boldsymbol{v}||) \log n)$ *bit operations. The output will satisfy* $||\boldsymbol{K}|| \leq ||\boldsymbol{v}||^2$*. The number of nonzero entries in* $\boldsymbol{K}$ *will be bounded by* $n(1 + \log_2 n)$*.*

Proof. Suppose $\boldsymbol{v}$ has m nonzero entries. Let $\boldsymbol{P}$ be an $n \times n$ permutation matrix such that

$$\boldsymbol{P}\boldsymbol{v} = \left[\begin{array}{c} \bar{\boldsymbol{v}} \\ \hline \\ \end{array}\right] \in \mathbb{Z}^{n\times 1}$$

with all entries in $\bar{\boldsymbol{v}} \in \mathbb{Z}^{m\times 1}$ nonzero. Use Algorithm `SparseKernelBasis` to compute a kernel basis $\bar{\boldsymbol{K}}$ for $\bar{\boldsymbol{v}}$. Then

$$\left[\begin{array}{c|c} \bar{\boldsymbol{K}} & \\ \hline & \boldsymbol{I}_{n-m} \end{array}\right] \boldsymbol{P}$$

is a kernel basis for $\boldsymbol{v}$. The result now follows from Lemmas 16–18. □

We can now state the main result of the paper.

Theorem 20. *There exists an algorithm that takes as input a full column rank* $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$*, and returns as output a* $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ *such that* $\boldsymbol{C}\boldsymbol{A}$ *is a basis for* $\boldsymbol{A}$*. The output will satisfy* $||\boldsymbol{C}|| \leq n^4||\boldsymbol{A}||^2$ *and the number of nonzero entries in* $\boldsymbol{C}$ *will be bounded by* $n(1 + \log_2 n)$*. The cost of the algorithm is* $O(n^\omega\,\mathrm{B}(d)(\log n)^2)$ *bit operations, where* $d = \log n + \log ||\boldsymbol{A}||$*.*

Proof. Use the algorithm supporting Theorem 14 (Augmentation vector) to compute a $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ such that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$. Use the algorithm supporting Theorem 19 (Sparse kernel basis) to compute a left kernel $\boldsymbol{C}$ of $\boldsymbol{v}$. Return $\boldsymbol{C}$. Correctness follows from Corollary 6.

By Theorem 14, $\boldsymbol{v}$ satisfies $||\boldsymbol{v}|| \leq n^2||\boldsymbol{A}||$. The claimed bound for $||\boldsymbol{C}||$ and the bound on the number of nonzero entries in $\boldsymbol{C}$ follow from Theorem 19. The runtime bound follows from Theorems 14 and 19. □

8 CONCLUSIONS

Given a full column rank $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, we have given a deterministic algorithm that produces a $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ such that $\boldsymbol{C}\boldsymbol{A}$ is a basis for $\boldsymbol{A}$. Note that any such $\boldsymbol{C}$ must have at least $n - 1$ nonzero entries, otherwise it would be rank deficient. We show that the $\boldsymbol{C}$ produced by our algorithm has at most $n(1 + \log_2 n)$ nonzero entries, a logarithmic factor away from the lower bound.

A natural direction for further research is to extend the approach to a matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-k)}$ of full column rank $n - k$. Using the idea in the first part of the proof of Theorem 12, we may assume, up to permuting the rows of $\boldsymbol{A}$, that we can write $\boldsymbol{A}$ as

$$\boldsymbol{A} = \left[\begin{array}{c} \bar{\boldsymbol{A}} \\ \hline \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_k \end{array}\right]$$

where $\bar{\boldsymbol{A}}$ is nonsingular. Then k applications of the algorithm supporting Theorem 20 can be used to produce a $\boldsymbol{C}$ such that $\boldsymbol{C}\boldsymbol{A}$ is a basis for $\boldsymbol{A}$.

Initialize $\boldsymbol{S}_0 = \bar{\boldsymbol{A}}$. For $i = 1, 2, \ldots, k$ in succession, let

$$\bar{\boldsymbol{S}}_i := \left[\begin{array}{c} \boldsymbol{S}_{i-1} \\ \hline \boldsymbol{a}_i \end{array}\right]$$

and use Theorem 20 to find a $\bar{\boldsymbol{C}}_i$ such that $\boldsymbol{S}_i := \bar{\boldsymbol{C}}_i\bar{\boldsymbol{S}}_i$ is a basis for $\bar{\boldsymbol{S}}_i$. To combine the $\bar{\boldsymbol{C}}_i$, set

$$\boldsymbol{C}_i = \left[\begin{array}{c|c} \bar{\boldsymbol{C}}_i & \\ \hline & \boldsymbol{I}_{k-i} \end{array}\right] \in \mathbb{Z}^{(n-i)\times(n-i+1)}$$

for $1 \leq i \leq k$. The final transforming matrix is then given by

$$\boldsymbol{C} = \boldsymbol{C}_k\boldsymbol{C}_{k-1}\ldots\boldsymbol{C}_1.$$

If $k = O(1)$, then the $\boldsymbol{C}$ produced will still satisfy $\log ||\boldsymbol{C}|| \in O(\log n + \log ||\boldsymbol{A}||)$, but is no longer guaranteed to be sparse. For non-constant k, further ideas will be required.

REFERENCES

[1] S. Birmpilis, G. Labahn, and A. Storjohann. Deterministic reduction of integer nonsingular linear system solving to matrix multiplication. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'19*, page 58–65, New York, NY, USA, 2019. ACM.

[2] S. Birmpilis, G. Labahn, and A. Storjohann. A Las Vegas algorithm for computing the Smith form of a nonsingular integer matrix. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'20*, page 38–45, New York, NY, USA, 2020. ACM.

[3] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3rd edition, 2013.

[4] S. Gupta, S. Sarkar, A. Storjohann, and J. Valeriote. Triangular x-basis decompositions and derandomization of linear algebra algorithms over $\mathsf{K}[x]$. *Journal of Symbolic Computation*, 47(4), 2012. Festschrift for the 60th Birthday of Joachim von zur Gathen.

[5] O. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3:45–56, 1982.

[6] J. Li and P. Q. Nguyen. Computing a lattice basis revisited. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'19*, page 275–282, New York, NY, USA, 2019. ACM.

[7] N. J. A. Sloane and The OEIS Foundation Inc. Entry a033156 in the on-line encyclopedia of integer sequences, 2020. URL http://oeis.org/A033156.

[8] A. Storjohann. *Algorithms for Matrix Canonical Forms*. PhD thesis, Swiss Federal Institute of Technology, ETH–Zurich, 2000.

[9] A. Storjohann. The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity*, 21(4):609–650, 2005. Festschrift for the 70th Birthday of Arnold Schönhage.