Computing bases in Hermite normal form of lattices of integer relations

George Labahn

Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada N2L 3G1

Arne Storjohann

Cheriton School of Computer Science, University of Waterloo, Waterloo ON, Canada N2L 3G1

Abstract

Given a full column rank $M \in \mathbb{Z}^{\ell \times m}$ and an $F \in \mathbb{Z}^{n \times m}$ we present an algorithm to compute the $n \times n$ basis in Hermite form of the integer lattice comprised of all rows $p \in \mathbb{Z}^{1 \times n}$ such that $pF \in \mathbb{Z}^{1 \times m}$ is in the integer lattice generated by the rows of M. The algorithm is randomized of the Las Vegas type, that is, it can fail with probability at most $1/2$, but if fail is not returned it guarantees to produce the correct result. When M is square and $F = I_m$, then the computed basis is the Hermite normal form of M, and the algorithm uses about the same number of bit operations as required to multiply together two matrices of the same dimension and size of entries as M.

Keywords: Hermite normal form, Integer relations lattice, Howell form, Smith massager,
MSC codes: 68Q25, 68W30, 15A21, 15A36

1. Introduction

There has been considerable recent work on designing algorithms for computations on integer (and polynomial) matrices that have about the same cost as multiplying together two matrices of the same dimension and size of entries as the input matrix. For a nonsingular $m \times m$ integer input matrix M the target complexity is $(m^\omega \log ||M||)^{1+o(1)}$ bit operations, where ω is the exponent of matrix multiplication, $||M|| = \max_{ij} |M_{ij}|$ denotes the largest entry in absolute value, and the "$+o(1)$" in the exponent indicates a missing factor $c_1(\log n)^{c_2}(\log\log ||M||)^{c_3}$ for positive real constants c_1, c_2, c_3. This complexity has been achieved with a deterministic algorithm for computing the rational solution to a linear system $\vec{x}M = \vec{b}$ [8], and with Las Vegas randomized algorithms for computing the diagonal Smith normal form S of M [5] and the Smith form with unimodular multipliers $MV = US$ [6].

However, no previous algorithm with the desired complexity exists for computing the Hermite normal form of an integer matrix. Recall that a full column rank $\ell \times m$ matrix

$$\begin{bmatrix} h_1 & h_{12} & \cdots & h_{1m} \\ & h_2 & \cdots & h_{2m} \\ & & \ddots & \vdots \\ & & & h_m \\ & & & \end{bmatrix}$$

is in (row) Hermite form if all its entries are nonnegative, the off-diagonal entries h_{*j} are strictly smaller than the diagonal entry h_j in the same column and any zero rows are at the bottom. For every full column rank matrix $M \in \mathbb{Z}^{\ell \times m}$, there is a unimodular matrix $W \in \mathbb{Z}^{\ell \times \ell}$ such that $H = WM$ is in Hermite form. The

Email addresses: glabahn@uwaterloo.ca (George Labahn), astorjoh@uwaterloo.ca (Arne Storjohann)

form is unique with its existence dating back to 1851 [15]. The first m rows of H, which we call the Hermite basis of M, gives a canonical presentation for $\mathcal{L}(M)$, the lattice generated by the $\mathbb{Z}$-linear combinations of the rows of M.

1.1. Main contributions

In this paper we present a new algorithm for computing the Hermite form of a full column rank input matrix $M \in \mathbb{Z}^{\ell \times m}$ using $(\ell m^{\omega-1} \log ||M||)^{1+o(1)}$ bit operations. The algorithm is Las Vegas randomized, that is, it can report FAIL with probability at most $1/2$, and if FAIL is not reported, then the output is guaranteed to be correct. More precisely we have:

Theorem 1. *Let $M \in \mathbb{Z}^{\ell \times m}$ have full column rank. There exists a Las Vegas randomized algorithm that computes the Hermite form of M in $O(\ell m^{\omega-1}\, \mathsf{B}(D/m + \log m)(\log m)^2))$ bit operations, where D is a bound for the sum of the bitlengths of the columns of M.*

Here, $\mathsf{B}(t)$ bounds the cost of operations on integers with bitlength bounded by t, including gcd-related operations, and by "bitlength of a column" we mean the bitlength of the largest entry in absolute value. The term D/m in the cost estimate is thus a bound for the average column bitlength of M. We always have $D/m \in O(\log ||M||)$. Section 1.4 gives our cost model.

Our approach is to solve the more general problem of finding the *Hermite bases of an integer relations lattice*. Given $M \in \mathbb{Z}^{\ell \times m}$ having full column rank and $F \in \mathbb{Z}^{n \times m}$, the set

$$\mathcal{R}(M, F) := \left\{ p \in \mathbb{Z}^{1 \times n} \mid pF = \mathbf{0} \bmod M \right\} \tag{1}$$

defines an integer relations lattice for F with modulus M. Here the notation $A = \mathbf{0} \bmod M$ stands for "$A = QM$ for some integer matrix Q," that is, the rows of A are in the lattice generated by the rows of M. A basis for the lattice $\mathcal{R}(M, F)$ is any nonsingular matrix $P \in \mathbb{Z}^{n \times n}$ that has minimal determinant in absolute value among all matrices that satisfy $PF = \mathbf{0} \bmod M$. In this paper we present an algorithm for computing the unique basis for $\mathcal{R}(M, F)$ that is in Hermite form.

Theorem 2. *Let $M \in \mathbb{Z}^{\ell \times m}$ have full column rank and $F \in \mathbb{Z}^{n \times m}$. There exists a Las Vegas randomized algorithm that computes the Hermite basis H of $\mathcal{R}(M, F)$ in*

$$O((\ell + n) m^{\omega-1}\, \mathsf{B}(D/m + \log m)(\log m)^2)$$

plus

$$O(n^{\omega}\, \mathsf{B}(D/n + \log n)(\log n)^2)$$

bit operations, where D is a bound for the sum of the bitlengths of the columns of M and F.

As an example consider when $F = I_m$ in (1). Then $\mathcal{R}(M, I)$ is equal to $\mathcal{L}(M)$ and thus the problem of computing the Hermite basis of $\mathcal{R}(M, I)$ is precisely that solved by Theorem 1. As a second example, if $M \in \mathbb{Z}^{m \times m}$ is a diagonal matrix of arbitrary nonzero moduli, then the lattice of solutions to the linear diophantine system $\vec{x}A = \vec{b} \bmod M$ is given by the integer relations lattice

$$\mathcal{R}\left(M, \begin{bmatrix} -\vec{b} \\ A \end{bmatrix}\right).$$

The Hermite basis of this lattice gives a compact presentation of the general solution to the system. Intersections of integer lattices can also be be conveniently described by integer relations lattices.

1.2. Historical background

The last few decades has seen considerable progress in reducing the cost of computations on polynomial matrices over $\mathsf{K}[x]$. Over $\mathsf{K}[x]$ the complexity counts the number of required field operations from K, and instead of $\log ||M||$ for integer matrices the measure of size is a bound on the degrees of entries in the input matrix. For polynomial matrices there are deterministic reductions to matrix multiplication for the problems

linear solving [13], nullspace/kernel [41], column bases [40], minimal or order basis [39], minimal interpolation bases [20], determinant and Hermite normal forms [23]. In the case of polynomial Smith forms there is a Las Vegas randomized algorithm [33].

Effective computation of Hermite normal forms of integer and polynomial matrices has a long history. Procedures developed during the 1970–1990's produced algorithms that were first provably polynomial time [21]. In the case of integer matrices a series of papers [9, 10, 18, 14] reduced the complexity to $(n^4 \log ||M||)^{1+o(1)}$ bit operations. Matrix multiplication was incorporated later [35, 31] to get algorithms having a worst case complexity $(n^{\omega+1} \log ||M||)^{1+o(1)}$. These algorithms, although presented for integer matrices, could also be modified to work for polynomial matrices having similar complexities with size measured by degrees. Later a series of techniques based on heuristic methods, for example found in [25, 29, 28, 24], were presented which reduced this complexity in some important cases. However, these heuristic algorithms also require strong assumptions, for example, that there only be a small number of non-trivial ($\neq 1$) late diagonal entries of the Hermite form. Most recently, a Las Vegas randomized algorithm [7] was given having complexity $(n^3 \log ||M||)^{1+o(1)}$ for any nonsingular M.

Relation lattices $\mathcal{R}(M, F)$ have mostly previously appeared for M and F being matrices of polynomials $\mathsf{K}[x]$. For example, Padé approximants and more generally Hermite-Padé approximants are included when $M = x^N$, while simultaneous Padé approximants are included when M is a diagonal matrix of powers of x. Rational interpolants are included when M is a diagonal matrix of linear factors describing interpolation conditions. Finding bases for the $\mathsf{K}[x]$ modules is useful as it describes all possible approximants or interpolants for a given rational expression. This includes, for example, descriptions of the well-known block structures of Padé and multi-point Padé approximation tables [3]. Fast algorithms for basis computation in these cases include those in [2] and [26].

1.3. Our approach and an outline of the paper

We view the problem of computing Hermite forms in terms of finding *minimal denominators* of matrices of rational numbers. If H is the Hermite form of a nonsingular integer matrix M then HM^{-1} is unimodular over $\mathbb{Z}$. It follows that $\det H$ is minimal in absolute value among all nonsingular integer matrices that clear the denominators of M^{-1} upon premultiplication. Examples and some basic properties of minimal denominators can be found in [7, Section 3.1].

Instead of working with M^{-1} explicitly we use a compact approximation of the form FS^{-1}, where F is an integer matrix and S is the Smith form of M. The pair (S, F) is called a *Smith massager* for M, and was introduced in [6, Definition 1] for efficiently computing Smith forms. Here we use the fact that H is the Hermite basis of $\mathcal{R}(S, F)$. Section 2.2 gives the background and results on Smith massagers needed in this paper.

A Smith massager (S, F) has two important properties. First, S and F are right coprime as integer matrices, which is equivalent to having $\det S = \det H$, where H is the Hermite basis of $\mathcal{R}(S, F)$. Second, because S is diagonal its modulus action in the equation $HF = \mathbf{0} \bmod S$ is decoupled with respect to the columns, which means that we can assume without loss of generality that (S, F) is a *reduced* Smith massager, with the entries in each column of F having magnitude strictly less than the corresponding diagonal entries of S. These properties ensure that the total size in bits required to write down (S, F) is bounded in the worst case by that required to write down the Hermite basis H of $\mathcal{R}(S, F)$.

Our main result is a recursive algorithm to compute the Hermite basis corresponding to a Smith massager. During the course of the algorithm integer relations lattices with a description $\mathcal{R}(M, G)$ can arise that are not necessarily in the form that we require. In particular, M may have more rows than columns, M may not be in Smith form, and M and G may not be coprime. In order to handle such an arbitrary input we show how to construct from (M, G) a Smith massager (S, F) for the Hermite basis of $\mathcal{R}(M, G)$. The construction of (S, F) uses a sequence of transformations:

$$(M, G) \xrightarrow{1} \left(\begin{bmatrix} M_1 \\ \hline M_2 \end{bmatrix}, G \right) \xrightarrow{2} \left(\begin{bmatrix} S_1 \\ \hline M_3 \end{bmatrix}, G_1 \right) \xrightarrow{3} (T_1, G_1) \xrightarrow{4} (S_2, G_2) \xrightarrow{5} (K, C) \xrightarrow{6} (S, F). \tag{2}$$

Transformations 1–6 achieve, in turn, that M_1 is a nonsingular submatrix of M, S_1 is the Smith form of M_1, T_1 is the Hermite basis of $\mathcal{L}(S_1) + \mathcal{L}(M_3)$ (the lattice generated by the rows of S_1 and M_3), S_2 is the Smith

form of T_1, (K, C) has inputs which are coprime as integer matrices, and S is the Smith form of K. Section 3 explains in detail these transformations and establishes their correctness, that is, that the transformed input generates the same lattice of integer relations.

The transformations mentioned above imply that computing a Hermite basis for $\mathcal{R}(M, G)$ reduces to finding a Hermite basis for $\mathcal{R}(S, F)$ where (S, F) is a Smith massager. The main idea of our recursive algorithm for computing the Hermite basis of $\mathcal{R}(S, F)$ is to split into two subproblems based on the columns of the output basis H. The algorithm uses a parameter m that is an upper bound on the number of nontrivial columns (with diagonal entry > 1) in the Hermite basis being computed. If H is $n \times n$ then at the top level of the recursion we have $m = n$. For any partition $m = m_1 + m_2$ we can uniquely factor $H = H_2 H_1$ as

$$H = H_2 H_1 = \begin{bmatrix} I_{m_1} & H_{12} \\ & \bar{H}_2 \end{bmatrix} \begin{bmatrix} \bar{H}_1 & \\ & I_{m_2} \end{bmatrix} = \begin{bmatrix} \bar{H}_1 & H_{12} \\ & \bar{H}_2 \end{bmatrix}. \tag{3}$$

Combining these two factors into a single Hermite basis is free. The difficulty in this case is defining the two subproblems. In Section 4 we show how to use a subset of the transformations in (2) to efficiently construct Smith massagers for H_1 and H_2, resulting in an algorithm for computing H.

The computations providing the biggest challenge for finding Smith massagers for each of the two subproblems of our algorithm are transformations 3 and 5. For transformation 3 note that the input matrix

$$\left[\begin{array}{c} S_1 \\ \hline M_3 \end{array}\right] \quad \text{has Hermite form} \quad \left[\begin{array}{c} T_1 \\ \hline \end{array}\right], \tag{4}$$

where T_1 is the Hermite basis of $\mathcal{L}(S_1) + \mathcal{L}(M_3)$, as required. It turns out that the presence of S_1 means that T_1 can be recovered by computing the row *Howell form* of this input matrix over $\mathbb{Z}/(s)$, with s being the largest invariant factor of S_1. A fast algorithm exists for the Howell form, and working modulo s can already give an improved bit complexity compared to using a "modulo determinant" Hermite form algorithm, which for this input matrix could work modulo $\det S_1$. For transformation 5 a similar approach also works. If T_2 is the Hermite basis of $\mathcal{L}(S_2) + \mathcal{L}(G_2)$, then

$$\begin{bmatrix} T_2 & & I \\ G_2 & I & \\ S_2 & & \end{bmatrix} \quad \text{has Hermite form with shape} \quad \begin{bmatrix} T_2 & & * \\ & I & C \\ & & K \end{bmatrix}, \tag{5}$$

and $\mathcal{R}(S_2, G_2) = \mathcal{R}(K, C)$ with the latter having coprime inputs. The block T_2 which is needed to complete the defintion of the input matrix in (5) can be computed by transforming to Hermite form of an input matrix of the type shown in (4). To compute the Hermite form in (5) we can work modulo the largest invariant factor of S_2. This is described in Section 5 along with the cost of each transformation. Note that C and K will have good bounds on their total size because they are blocks of the Hermite form an input matrix that has determinant equal in absolute value to $\det S_2$.

In the worst case, the largest invariant factor s of a Smith form S can have $\log s \in \Omega(\log \det S)$, and existing algorithms to compute the Hermite forms in (4) and (5) are still too costly for our complexity goal. In Section 6 we develop a new *slicing* algorithm that more fully exploits the presence and structure of S and achieves a cost that depends on the *average* bitlength $O((\log \det S)/m)$, where m is the dimension of S.

In addition to transformations 3 and 5, the main other computational step needed for finding the Smith massagers for each of the two subproblems of our algorithm is to transform a modulus to Smith form. In the case of transformation 2 we use an existing algorithm to compute a Smith massager (S_1, V_1) for M_1 and then set $M_3 = \mathbf{c}\mathrm{mod}(M_2 V_1, S_1)$ and $G_1 = \mathbf{c}\mathrm{mod}(G V_1, S_1)$, where $\mathbf{c}\mathrm{mod}(\cdot, S)$ denotes matrix multiplication modulo a diagonal matrix S. Transformations 4 and 6 work the same way, each requiring one Smith massager computation and one **c**mod matrix multiplication. Beyond computing Smith massagers, all of the computational effort to achieve the transformations discussed so far is integer matrix multiplication, albeit of a special type. In Section 7 we develop procedures which use partial linearization to do these **c**mod matrix multiplications in the required bit complexity.

The total cost of our Hermite basis algorithm depends on the dimension parameters n (which is fixed) and m, and the precision parameter $\det H$. The algorithm constructs the subproblems using a factorization

(3) with $m_1 = m_2 = m/2$, and $(\det H_1)(\det H_2) = \det H$. In Section 8 we give a complexity analysis that shows that this even partitioning of m but output dependent splitting of $\det H$ corresponds to a recursion tree that is root-heavy.

The details needed for the case where M is rectangular but of full column rank requires a different precision measure and is described in Section 9. In Section 10 we give a number of examples solved by our work. This includes solving a version of the multivariable Chinese remainder problem and efficient computation of the Hermite form for a number of interesting cases. The paper ends with a conclusion and topics for future research.

1.4. Cost model

The number of bits in the binary representation of an integer a is

$$\text{bitlength}(a) := \begin{cases} 1 & \text{if } a = 0 \\ 1 + \lfloor \log_2 |a| \rfloor & \text{otherwise} \end{cases}.$$

We will use the fact that $|a| < 2^{\text{bitlength}(a)}$. Now let $A \in \mathbb{Z}^{n\times m}$. We define $\text{bitlength}(A) := \text{bitlength}(||A||)$, the bitlength of the largest entry in absolute value. Cost estimates will often be expressed in terms of a bound D for the sum of the bitlengths of the columns of A. Note that if A is the zero matrix then we can take $D = m$. If A has entries in each column of magnitude less than the corresponding diagonal entries of a nonsingular diagonal matrix $S \in \mathbb{Z}_{\geq 0}^{m\times m}$ then we can take $D = m + \log_2 \det S$.

Our cost estimates are given in terms of a fixed $2 < \omega \leq 3$ such that two $n \times n$ matrices can be multiplied together in $O(n^\omega)$ operations from a commutative ring using only the operations $\{+, -\times\}$. The standard algorithm for matrix multiplication has $\omega = 3$, while the current best known asymptotic upper bound for ω allows $\omega < 2.37286$ [1].

Following [12, Section 8.3], we give cost estimates using a nondecreasing function $\mathsf{M}(d) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ together with an algorithm that can multiply two integers with magnitude bounded by $a \geq 1$ using at most $\mathsf{M}(\log a)$ bit operations. We assume $\mathsf{M}(d) = 1$ for all $0 \leq d \leq 1$. We make the following assumptions in order to simplify cost estimates in the analysis of our algorithms.

- $\mathsf{M}(d_1) + \mathsf{M}(d_2) \leq \mathsf{M}(d_1 + d_2)$ for all $d_1, d_2 \geq 1$ (superlinearity).
- $\mathsf{M}(d_1 d_2) \leq \mathsf{M}(d_1)\mathsf{M}(d_2)$ for all $d_1, d_2 \geq 0$ (submultiplicativity).
- $\mathsf{M}(d) \in O(d^{\omega-1-\epsilon_0})$ for some $0 < \epsilon_0 < \omega - 2$.

The last assumption states that if fast matrix multiplication techniques are used, then fast integer multiplication should be used as well. We use $\mathsf{B}(d)$ to bound the cost of integer gcd-related computations such as the extended euclidean algorithm. We can always take $\mathsf{B}(d) = O(\mathsf{M}(d) \log d)$. The assumptions stated above for M also apply to B.

2. Preliminaries

In this section we define notation and basic concepts, and recall from the literature some computational subroutines that we need.

2.1. Basic notation and concepts

We identify the residue class ring $\mathbb{Z}/(s)$ with the set $\{0, 1, \ldots, s-1\}$. Let $S \in \mathbb{Z}_{\geq 0}^{m\times m}$ be diagonal and $A \in \mathbb{Z}^{*\times m}$. By $\mathbf{c}\text{mod}(A, S)$ we mean the matrix obtained from A by replacing the entries in each column with their residues modulo the corresponding diagonal entries in S. If $A = \mathbf{c}\text{mod}(A, S)$ then we say that A *is reduced column-modulo* S. The notations $\mathbf{r}\text{mod}(*, S)$ and *reduced row-modulo* S are analogous, but for rows.

Let A be a full column rank integer matrix. The set $\mathcal{L}(A)$ of all $\mathbb{Z}$-linear combinations of rows of A forms a lattice. By *basis of* $\mathcal{L}(A)$ we mean a nonsingular integer matrix P with $\mathcal{L}(P) = \mathcal{L}(A)$. We will often simply

say *basis of* A to mean basis of $\mathcal{L}(A)$. By *Hermite basis of* A we mean the basis of A that is in Hermite form. Note that if A is square then the Hermite basis of A and the Hermite form of A are the same. Otherwise, the Hermite basis of A is simply the submatrix of the Hermite form of A consisting of the nonzero rows.

Let $H \in \mathbb{Z}^{n\times n}$ be a Hermite basis. We say that H *is index* (k, m) if it has the shape

$$H = \begin{bmatrix} I_k & * & \\ & \bar{H} & \\ & & I_{n-k-m} \end{bmatrix} \in \mathbb{Z}^{n\times n}, \tag{6}$$

where $\bar{H}$ has m columns, and where integers k and m satisfy $0 \le k \le k+m \le n$. Note that every Hermite basis of dimension n is index $(0, n)$ since no additional structure is imposed.

Let M and F be integer matrices with the same column dimension. If

$$\begin{bmatrix} M \\ F \end{bmatrix} \tag{7}$$

has full column rank, then we say that M *and* F *are coprime* if the only common integer matrix right factor is unimodular. A necessary and sufficient condition for M and F to be coprime is that the Hermite basis of $\mathcal{L}(M) + \mathcal{L}(F)$, which is the Hermite basis of (7), is equal to I.

2.2. Smith massagers

As in [7] we view the Hermite form H of a nonsingular matrix M as the unique minimal denominator of its rational inverse M^{-1}. The key tool used to work efficiently in this setting is the notion of a Smith massager, introduced in [5, 6] for computing the Smith form and multiplier matrices for a nonsingular matrix.

Definition 3 ([6, Definition 1])**.** *Let* $M \in \mathbb{Z}^{n\times n}$ *be nonsingular with Smith form* S*. The pair of matrices* (S, F) *is a* Smith massager *for* M *if* $F \in \mathbb{Z}^{n\times n}$ *and*

(i) $MF \equiv 0 \bmod S$*, and*

(ii) there exists a matrix $W \in \mathbb{Z}^{n\times n}$ *such that* $WF \equiv I_n \bmod S$*.*

Note that if (S, F) is a Smith massager for a matrix M, then we can replace F with $\mathbf{c}\mathrm{mod}(F, S)$. In this case we say that (S, F) is *reduced*.

Remark 4. *Note that condition (ii) in Definition 3 is equivalent to* S *and* F *being coprime.*

Remark 5. *A Smith massager was originally defined to be the matrix* F *alone. For convenience, since the algorithm we use to compute a Smith massager will also compute* S*, we define the Smith massager to be the pair* (S, F)*.*

The key feature of a Smith massager that we exploit here is the following.

Lemma 6. *Let* $M \in \mathbb{Z}^{n\times n}$ *be nonsingular with Smith form* S*. Then any Smith massager* (S, F) *for* M *has the property that* FS^{-1} *has minimal denominator* M*.*

Proof. This follows from [6, Theorem 4] where it is shown that the lattices

$$\{v \in \mathbb{Z}^{1\times n} \mid vM^{-1} \in \mathbb{Z}^{1\times n}\} \text{ and } \{v \in \mathbb{Z}^{1\times n} \mid vFS^{-1} \in \mathbb{Z}^{1\times n}\}$$

are identical. □

As currently defined, a Smith massager (S, F) requires F to be square. However, the initial columns of F corresponding to trivial diagonal elements do not play a role. In particular, if (S, F) is reduced and S has $n - m$ trivial diagonals, then we can decompose S and F as $S = \mathrm{diag}(I_{n-m}, \hat{S})$ and $F = \left[\begin{array}{c|c} \mathbf{0} & \hat{F} \end{array}\right]$. The lattices

$$\{v \in \mathbb{Z}^{1\times n} \mid vFS^{-1} \in \mathbb{Z}^{1\times n}\} \text{ and } \{v \in \mathbb{Z}^{1\times n} \mid v\hat{F}\hat{S}^{-1} \in \mathbb{Z}^{1\times m}\}$$

are identical and hence have the same Hermite basis.

Example 7. *Consider the matrix*

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix}$$

with Smith form $\operatorname{diag}(1, 1, 24)$. *A reduced Smith massager* (S, F) *for* M *is given by*

$$S = \begin{bmatrix} 24 \end{bmatrix} \text{ and } F = \begin{bmatrix} 19 \\ 10 \\ 3 \end{bmatrix}.$$

Indeed, the conditions of Definition 3 are satisfied since

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 1 \end{bmatrix} \begin{bmatrix} 19 \\ 10 \\ 3 \end{bmatrix} = \begin{bmatrix} 48 \\ 144 \\ 216 \end{bmatrix} \equiv \mathbf{0} \bmod 24 \quad \text{and} \quad \mathcal{L}(\begin{bmatrix} 24 \end{bmatrix}) + \mathcal{L}\left(\begin{bmatrix} 19 \\ 10 \\ 3 \end{bmatrix} \right) = \mathcal{L}(\begin{bmatrix} 1 \end{bmatrix}).$$

The Hermite basis of the lattice $\{v \in \mathbb{Z}^{1\times 3} \mid vF \equiv 0 \bmod 24\}$ *is*

$$H = \begin{bmatrix} 1 & 2 & 3 \\ & 3 & 6 \\ & & 8 \end{bmatrix},$$

which is equal to the Hermite form of M.

A reduced Smith massager (S, F) for a Hermite basis H does not require more space to store than H itself in the worst case. In particular, if H has dimension m then it can be represented using $O(m \log \det H)$ bits by storing only the nontrivial columns [7, Lemma 13]. Since $\det S = \det H$ the same bound holds for representing (S, F).

Theorem 8 ([5, 6])**.** *There exists a randomized algorithm* `SmithMassager`$[\epsilon](M)$ *that takes as input a nonsingular* $M \in \mathbb{Z}^{m\times m}$*, and returns as output a reduced Smith massager* (S, F) *for* M *with cost* $O(m^\omega \, \mathsf{B}(D/m + \log m)(\log m)^2 \log(1/\epsilon))$ *bit operations, where* D *is a bound for the sum of the bitlengths of the columns of* M*. The algorithm is probabilistic of the Las Vegas type: it either returns a correct result, or reports* FAIL *with probability at most* ϵ.

Proof. The statement of the theorem is slightly modified from [6, Theorem 19], which was specialized to the case $\epsilon = 1/2$, and which gave a cost estimate with the factor $\mathsf{B}(\log ||M|| + \log m)$. To obtain the result in the current theorem we can iterate that algorithm $\log_2(1/\epsilon)$ times to reduce the chance of returning FAIL to ϵ, and use partial linearization [6, Theorem 36] to replace the maximum column bitlength $\log ||M||$ with the average column bitlength D/m. □

2.3. Computation of some Hermite forms using modular arithmetic

It is well known that one can compute the Hermite form of a nonsingular matrix $A \in \mathbb{Z}^{n\times n}$ by working modulo $|\det A|$ to control intermediate expression swell (see, for example, [10, 18, 14, 35]). In some cases one can work modulo a divisor $s \in \mathbb{Z}_{>0}$ of $\det A$ by passing over the Howell form [16], a natural generalization to $\mathbb{Z}/(s)$ of the Hermite form over $\mathbb{Z}$. The following result applies more generally to a full column rank input matrix.

Lemma 9 ([4, Part 2 of Lemma 7])**.** *Let* $A \in \mathbb{Z}^{n\times m}$ *and* $s \in \mathbb{Z}_{>0}$ *be such that* $sI \subseteq \mathcal{L}(A)$*. Then the canonical lifting of the Howell form of* A *over* $\mathbb{Z}/(s^2)$ *yields the Hermite form of* A *over* $\mathbb{Z}$.

Theorem 10. *Let* $A \in \mathbb{Z}^{n\times m}$ *and* $s \in \mathbb{Z}_{>0}$ *be such that* $sI \subseteq \mathcal{L}(A)$*. If* $\log ||A|| \in O(\log s)$*, then the Hermite basis* H *of* A *can be computed in* $O(nm^{\omega-1} \, \mathsf{B}(\log s))$ *bit operations.*

Proof. By Lemma 9 the Hermite basis H of A coincides with the first m rows of the Howell form of A over $\mathbb{Z}/(s^2)$, which can be computed in the allotted time [36, Section 4]. □

Example 11. *Let*

$$M_1 := \begin{bmatrix} A \\ S \end{bmatrix} \quad \text{and} \quad M_2 := \begin{bmatrix} T & & I \\ B & I & \\ S & & \end{bmatrix}$$

be integer matrices with S a nonsingular Smith form. Let s be the largest invariant factor of S. Then $sI = sS^{-1}S = S^*S$, with S^* an integer matrix and so $sI \in \mathcal{L}(M_1)$. Similarly

$$sI = sM_2{}^{-1}M_2 = \begin{bmatrix} & & S^* \\ & sI & -BS^* \\ sI & & -TS^* \end{bmatrix} \begin{bmatrix} T & & I \\ B & I & \\ S & & \end{bmatrix}$$

and so $sI \in \mathcal{L}(M_2)$. Thus we can find the Hermite forms of both M_1 and M_2 by computing their Howell forms over $\mathbb{Z}/(s^2)$. Computing the Hermite forms of matrices with these shapes are key steps in the Hermite basis algorithm presented in Section 4.

3. Bases of integer relations: simplifications and properties

In this section we show how a given integer relations lattice $\mathcal{R}(M, F)$ can be transformed to a new description that is in some way simpler, for example with M in Hermite form, or with M and F being coprime. These transformations allow us to control the dimension of the matrices arising during computations along with the bitlength of their entries.

3.1. Integer relations lattices

Given $M \in \mathbb{Z}^{\ell \times m}$ having full column rank and $F \in \mathbb{Z}^{n \times m}$, the set of integer vectors

$$\mathcal{R}(M, F) := \left\{ p \in \mathbb{Z}^{1 \times n} \mid pF = \mathbf{0} \bmod M \right\} \tag{8}$$

forms a lattice, which we call the *integer relations lattice* for F with *modulus* M. Here the notation $A = \mathbf{0} \bmod M$ stands for "$A = QM$ for some integer matrix Q," that is, the rows of A are in $\mathcal{L}(M)$. We have adopted the notation in (8) from [26] where it was introduced in the context of polynomial matrices.

Remark 12. *The definition of $\mathcal{R}(M, F)$ extends naturally to the case when M and F are over the reals. For example, P being a minimal denominator of a rational matrix R is equivalent to P being a basis for $\mathcal{R}(I, R)$. In this paper, however, we stipulate that M and F be integral.*

Lemma 13 gives alternate description of $\mathcal{R}(M, F)$ that is useful for Hermite basis computation.

Lemma 13. *$\mathcal{R}(M, F)$ corresponds to the full column rank input matrix*

$$\left[\begin{array}{c|c} M & \\ F & I \end{array}\right] \quad \text{that has Hermite basis} \quad \left[\begin{array}{c|c} T & * \\ & H \end{array}\right] \tag{9}$$

where $\mathcal{L}(T) = \mathcal{L}(M) + \mathcal{L}(F)$ and $\mathcal{L}(H) = \mathcal{R}(M, F)$.

Proof. That $\mathcal{L}(T) = \mathcal{L}(M) + \mathcal{L}(F)$ follows from the fact that the leading columns of the Hermite basis depend only the corresponding leading columns of the input matrix.

To establish that $\mathcal{L}(H) = \mathcal{R}(M, F)$ we will show that $\mathcal{L}(H) \subseteq \mathcal{R}(M, F)$ and $\mathcal{R}(M, F) \subseteq \mathcal{L}(H)$. From (9) we have that $\left[\begin{array}{c|c} & H \end{array}\right] = -Q \left[\begin{array}{c|c} M & \end{array}\right] + H \left[\begin{array}{c|c} F & I \end{array}\right]$ for some integer matrix Q, and thus $HF = QM$. This shows that $\mathcal{L}(H) \subseteq \mathcal{R}(M, F)$. Now let $\bar{H}$ be the Hermite basis of $\mathcal{R}(M, F)$. Then $\bar{H}F = \bar{Q}M$ for some integer matrix $\bar{Q}$ and it follows that $\left[\begin{array}{c|c} & \bar{H} \end{array}\right] = -\bar{Q} \left[\begin{array}{c|c} M & \end{array}\right] + \bar{H} \left[\begin{array}{c|c} F & I \end{array}\right]$. This shows that $\mathcal{L}(\bar{H})$, which is equal to $\mathcal{R}(M, F)$, is a subset of $\mathcal{L}(H)$. □

3.2. Transformations to simplify the inputs of $\mathcal{R}(M,F)$

In this subsection we describe a number of transformations of an input $\mathcal{R}(M,F)$. All are based on the following general lemma, which follows from the definition of a relations lattice. We remark that in part 3, R is allowed to be over $\mathbb{Q}$, provided that MR and FR remain integral.

Lemma 14. *$\mathcal{R}(M,F)$ is equal to*

1. *$\mathcal{R}(\bar{M},F)$ for any $\bar{M}$ with $\mathcal{L}(\bar{M}) = \mathcal{L}(M)$.*
2. *$\mathcal{R}(M, F+QM)$ for any Q over $\mathbb{Z}$.*
3. *$\mathcal{R}(MR, FR)$ for any nonsingular matrix R.*
4. $\mathcal{R}\left(\left[\begin{array}{c|c} M' & \\ & M \end{array}\right], \left[\begin{array}{c|c} \mathbf{0} & F \end{array}\right]\right)$ *for any full column rank M'.*
5. $\mathcal{R}\left(M_1, \begin{bmatrix} F \\ M_2 \end{bmatrix}\right)\begin{bmatrix} I \\ \mathbf{0} \end{bmatrix}$, *if* $M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$ *with M_1 having full column rank.*

Proof. As the first four parts follow directly from definitions we only include a proof for the last identity. For this case note first that for any $p \in \mathcal{R}(M,F)$ there is a $q = \left[\begin{array}{c|c} q_1 & q_2 \end{array}\right]$ such that

$$pF = qM = \left[\begin{array}{c|c} q_1 & q_2 \end{array}\right]\begin{bmatrix} M_1 \\ M_2 \end{bmatrix}, \quad \text{that is,} \quad \left[\begin{array}{c|c} p & -q_2 \end{array}\right]\begin{bmatrix} F \\ M_2 \end{bmatrix} = q_1 M_1.$$

Thus

$$\left[\begin{array}{c|c} p & -q_2 \end{array}\right] \in \mathcal{R}\left(M_1, \begin{bmatrix} F \\ M_2 \end{bmatrix}\right), \quad \text{that is,} \quad p \in \mathcal{R}\left(M_1, \begin{bmatrix} F \\ M_2 \end{bmatrix}\right)\begin{bmatrix} I \\ \mathbf{0} \end{bmatrix}.$$

A similar argument shows that $p \in \mathcal{R}\left(M_1, \begin{bmatrix} F \\ M_2 \end{bmatrix}\right)\begin{bmatrix} I \\ \mathbf{0} \end{bmatrix}$ also implies $p \in \mathcal{R}(M,F)$. □

We remark that Lemma 13 gives an alternate way to prove the above results. For example, for Lemma 14.5 the corresponding input matrices from (9) are

$$\mathcal{R}(M,F) \sim \begin{bmatrix} M_1 & \\ M_2 & \\ F & I \end{bmatrix} \quad \text{and} \quad \mathcal{R}\left(M_1, \begin{bmatrix} F \\ M_2 \end{bmatrix}\right) \sim \left[\begin{array}{cc||c} M_1 & & \\ F & I & \\ M_2 & & I \end{array}\right].$$

Lemma 14.5 then follows by noting that the Hermite form of the first input matrix is equal to the leading columns of the Hermite form of second input matrix.

3.2.1. Transforming the modulus to Hermite and Smith forms

Suppose T is the Hermite basis of the modulus M in $\mathcal{R}(M,F)$. Then compared to M, which may have more rows than columns, T is square and nonsingular and has the benefit of being a canonical presentation of $\mathcal{L}(M)$. Lemma 15 shows that we can replace M by T, and that F can be replaced by its *remainder* $\bar{F}$ *with respect to* T, the unique matrix that satisfies $\bar{F} = F \bmod T$ and is reduced column-modulo the diagonal entries of T. The new inputs T and $\bar{F}$ have controlled size, with $T \in \mathbb{Z}^{m\times m}$ and $\bar{F} \in \mathbb{Z}^{n\times m}$ requiring only $O(m \log \det T)$ and $O(n \log \det T)$ bits to represent, respectively (see [7, Section 3.3]).

Lemma 15. *The following hold:*

1. *$\mathcal{R}(M,F) = \mathcal{R}(T,F)$, where T is the Hermite basis of M.*

2. If T is a Hermite basis, then $\mathcal{R}(T,F) = \mathcal{R}(T,\bar{F})$, where

$$\left[\begin{array}{c|c} I & F \\ \hline & T \end{array}\right] \quad \textit{has Hermite form} \quad \left[\begin{array}{c|c} I & \bar{F} \\ \hline & T \end{array}\right].$$

Proof. Part 1 follows directly from Lemma 14.1. For part 2, note that the unique transformation to Hermite form has the shape

$$\left[\begin{array}{c|c} I & Q \\ \hline & I \end{array}\right] \left[\begin{array}{c|c} I & F \\ \hline & T \end{array}\right] = \left[\begin{array}{c|c} I & \bar{F} \\ \hline & T \end{array}\right],$$

for some integer matrix Q. Then $\bar{F} = F + QT$ and Lemma 14.2 gives that $\mathcal{R}(T,F) = \mathcal{R}(T,\bar{F})$. □

It is also possible to replace the modulus M with its Smith form. In this case, however, we also need to modify F appropriately.

Lemma 16. *If M is nonsingular, then $\mathcal{R}(M,F) = \mathcal{R}(S,FW)$ where (S,W) is a Smith massager for M.*

Proof. From [6, Theorem 4] we have that

$$\{v \in \mathbb{Z}^{1\times m} \mid v = \mathbf{0} \bmod M\} = \{v \in \mathbb{Z}^{1\times m} \mid vW = \mathbf{0} \bmod S\}. \tag{10}$$

Let n be the row dimension of F. The following equation recalls the definition of $\mathcal{R}(M,F)$, and then applies (10) with $v = pF$.

$$\begin{aligned} \mathcal{R}(M,F) &= \{p \in \mathbb{Z}^{1\times n} \mid pF = \mathbf{0} \bmod M\} \\ &= \{p \in \mathbb{Z}^{1\times n} \mid pFW = \mathbf{0} \bmod S\} \\ &= \mathcal{R}(S,FW). \quad \square \end{aligned}$$

For a rectangular modulus we have the following extension of Lemma 16.

Lemma 17. *If M_1 is nonsingular, then*

$$\mathcal{R}\left(\left[\begin{array}{c} M_1 \\ M_2 \end{array}\right], F\right) = \mathcal{R}\left(\left[\begin{array}{c} S \\ \mathbf{c}\bmod(M_2W, S) \end{array}\right], \mathbf{c}\bmod(FW, S)\right),$$

where (S,W) is a Smith massager for M_1.

Proof. We have

$$\begin{aligned} \mathcal{R}\left(\left[\begin{array}{c} M_1 \\ M_2 \end{array}\right], F\right) &= \mathcal{R}\left(M_1, \left[\begin{array}{c} F \\ M_2 \end{array}\right]\right)\left[\begin{array}{c} I \\ \mathbf{0} \end{array}\right] && (11) \\ &= \mathcal{R}\left(S, \left[\begin{array}{c} FW \\ M_2W \end{array}\right]\right)\left[\begin{array}{c} I \\ \mathbf{0} \end{array}\right] && (12) \\ &= \mathcal{R}\left(\left[\begin{array}{c} S \\ M_2W \end{array}\right], FW\right). && (13) \end{aligned}$$

Line (11) follows by applying Lemma 14.5 in the forward direction, (12) follows from Lemma 16, and (13) follows by applying Lemma 14.5 in the reverse direction. By Lemma 14.1 and 14.2, we can replace the matrices M_2W and FW in (13) with $\mathbf{c}\bmod(M_2W,S)$ and $\mathbf{c}\bmod(FW,S)$, respectively. □

Lemma 18 shows that we can reduce the column dimension of the arguments of $\mathcal{R}(S,F)$ by removing leading columns corresponding to trivial invariant factors in S. Conversely, we could start with $\mathcal{R}(\hat{S},\hat{F})$ and implicitly add some trivial invariant factors to $\hat{S}$ and corresponding zero columns to $\hat{F}$ to adjust the column dimension upward.

Lemma 18. *If $S = \mathrm{diag}(I,\hat{S})$ and $F = \left[\begin{array}{c|c} * & \hat{F} \end{array}\right]$ then $\mathcal{R}(S,F)$ is equal to $\mathcal{R}(\hat{S},\hat{F})$.*

Proof. Using Lemma 14.2 and 14.4 in succession gives $\mathcal{R}(S,F) = \mathcal{R}(S, \left[\begin{array}{c|c} \mathbf{0} & \hat{F} \end{array}\right]) = \mathcal{R}(\hat{S},\hat{F})$. □

3.2.2. Removing common right divisors

A relations lattice $\mathcal{R}(M, F)$ can be simplified by removing any common right matrix divisors. We say that $\mathcal{R}(M, F)$ *has coprime inputs* when T in the following lemma is I.

Lemma 19. *Let T be the Hermite basis of $\mathcal{L}(M) + \mathcal{L}(F)$. Then $\mathcal{R}(M, F)$ is equal to $\mathcal{R}(MT^{-1}, FT^{-1})$, with the latter having coprime inputs.*

Proof. Postmultiplying any full column rank integer matrix by the inverse of a basis for its row lattice yields an integer matrix whose rows generate I. Thus $\mathcal{R}(MT^{-1}, FT^{-1})$ has inputs which are coprime. In addition, from Lemma 14.3 we have that $\mathcal{R}(M, F) = \mathcal{R}(MT^{-1}, FT^{-1})$. □

An issue with using Lemma 19 directly is that the bitlength of entries in MT^{-1} and FT^{-1} may be large.[1] The following theorem gives an alternative method to construct coprime inputs.

Theorem 20. *Let T be the Hermite basis of $\mathcal{L}(M) + \mathcal{L}(F)$. Then*

$$\begin{bmatrix} T & & I \\ F & I & \\ M & & \end{bmatrix} \text{ has Hermite basis with shape } \begin{bmatrix} T & & * \\ & I & C \\ & & K \end{bmatrix} \tag{14}$$

and $\mathcal{R}(M, F)$ is equal to $\mathcal{R}(K, C)$, with the latter having coprime inputs.

Proof. Since FT^{-1} and MT^{-1} are integral the following transformation is unimodular:

$$\begin{bmatrix} I & & \\ -FT^{-1} & I & \\ -MT^{-1} & & I \end{bmatrix} \begin{bmatrix} T & & I \\ F & I & \\ M & & \end{bmatrix} = \begin{bmatrix} T & & I \\ & I & -FT^{-1} \\ & & -MT^{-1} \end{bmatrix}. \tag{15}$$

The Hermite basis of the matrix on the right hand side of (15) has the shape shown in (14). By Lemma 15 we have $\mathcal{R}(K, C) = \mathcal{R}(-MT^{-1}, -FT^{-1})$, which is equal to $\mathcal{R}(MT^{-1}, FT^{-1})$. The result now follows from Lemma 19. □

3.3. Additional properties of bases of a lattice of integer relations

Ensuring that $\mathcal{R}(S, F)$ has coprime inputs links the structure of the Hermite basis H of $\mathcal{R}(S, F)$ with the modulus S. Theorem 21 allows us to conclude that $\det H = \det S$. Corollary 22 notes that the number of nontrivial invariant factors of S is bounded by the number of nontrivial columns of H.

Theorem 21. *Assume the inputs to $\mathcal{R}(S, F)$ are coprime with S a nonsingular Smith form. Then, up to trivial invariant factors, the Smith form of any basis for $\mathcal{R}(S, F)$ is equal to S.*

Proof. Since all bases for $\mathcal{R}(S, F)$ are left equivalent, it will suffice to prove the result for the Hermite basis H of $\mathcal{R}(S, F)$. The result follows from Lemma 13 which observes that

$$\left[\begin{array}{c|c} S & \\ \hline F & I \end{array}\right] \text{ has Hermite form } \left[\begin{array}{c|c} I & * \\ \hline & H \end{array}\right]. \tag{16}$$

In particular, the matrices in (16) are equivalent to $\mathrm{diag}(I, S)$ and $\mathrm{diag}(I, H)$, respectively. □

Corollary 22. *Assume the inputs to $\mathcal{R}(S, F)$ are coprime with S a nonsingular Smith form. If the Hermite basis H of $\mathcal{R}(S, F)$ is index $(*, m)$, then S has at most m nontrivial invariant factors.*

Proof. If H is index (k, m) it can be written using a block decomposition as shown in (6). Given its structure, H is right equivalent to the block diagonal matrix $\mathrm{diag}(I_k, \bar{H}, I_{n-m-k})$, which has Smith form equal to $\mathrm{diag}(I_{n-m}, \bar{S})$, where $\bar{S}$ is the Smith form of the $m \times m$ matrix $\bar{H}$. The result now follows from Theorem 21. □

[1] [8, Example 10] gives an example of an $m \times m$ Hermite form T with $\log_2 ||T|| = 1$ and $\log ||T^{-1}|| = \Theta(m)$.

4. The Hermite basis algorithm

In this section we present a recursive method to compute the Hermite basis H for an $\mathcal{R}(S, F)$ with coprime inputs and modulus S a nonsingular Smith form. The precondition of coprime inputs implies that $\det S = \det H$, which will be important for the cost analysis. We remark that if (S, F) is a Smith massager for a nonsingular matrix A, then the algorithm here directly results in a recursive algorithm to compute the Hermite form of A.

The key idea of our approach is to recursively split the problem into two subproblems according to the desired output basis H. Because a Hermite basis H is upper triangular, we can choose a factorization $H = H_2H_1$ such that the product H_2H_1 avoids any computation. For example, when H is $n \times n$ and $n = n_1 + n_2$ is a given partition of n we have

$$H = \begin{bmatrix} \bar{H}_1 & H_{12} \\ & \bar{H}_2 \end{bmatrix} = \begin{bmatrix} I_{n_1} & H_{12} \\ & \bar{H}_2 \end{bmatrix} \begin{bmatrix} \bar{H}_1 & \\ & I_{n_2} \end{bmatrix} = H_2H_1. \tag{17}$$

At the top level of the recursion the algorithm will choose the partitioning $n = \lfloor n/2 \rfloor + \lceil n/2 \rceil$, so that each subproblem recovers about half the columns of the Hermite form. The two recursive subproblems are to compute first H_1 and then H_2. This divide and conquer approach is based on the following theorem, which holds generally for any factorization H_2H_1 of a basis.

Theorem 23. *If H_2 and H_1 are nonsingular integer matrices with H_2H_1 a basis for $\mathcal{R}(M, F)$, then*

1. *H_1 is a basis for*
$$\mathcal{R}\left(\begin{bmatrix} M \\ H_1F \end{bmatrix}, F \right). \tag{18}$$
2. *H_2 is a basis for $\mathcal{R}(M, H_1F)$.*

Proof. Our proof is based on the fact that any two bases (as nonsingular matrices) are left multiples of each other via a unimodular matrix, or equivalently, have the same minimal determinant. We will implicitly use the fact that H_1 and H_2 are nonsingular.

First consider part 1. From the definition of an integer relations lattice we have that $\mathcal{L}(H_1)$ is subset of (18). Thus, to show that H_1 is a basis for (18), it will suffice to show that $\det H_1 \mid \det T_1$, where T_1 is any basis for (18). For any such T_1 there exist integer matrices Q_1 and Q_2 such that

$$T_1F = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} M \\ H_1F \end{bmatrix}. \tag{19}$$

Rewriting (19) gives

$$(T_1 - Q_2H_1)F = Q_1M. \tag{20}$$

Since H_2H_1 is a basis for $\mathcal{R}(M, F)$, from (20) we have $\mathcal{L}(T_1 - Q_2H_1) \subseteq \mathcal{L}(H_2H_1)$, so there exists a matrix G such that $T_1 - Q_2H_1 = G(H_2H_1)$. Solving for T_1 gives $T_1 = (GH_2 + Q_2)H_1$, and hence $\det H_1 \mid \det T_1$.

Now consider part 2. Because H_2H_1 is a basis for $\mathcal{R}(M, F)$, we have $\mathcal{L}(H_2) \subseteq \mathcal{R}(M, H_1F)$. Thus, to show that H_2 is a basis for $\mathcal{R}(M, H_1F)$, it will suffice to show that $\det H_2 \mid \det T_2$, where T_2 is a basis of $\mathcal{R}(M, H_1F)$. For such a basis T_2 we have $\mathcal{L}(T_2H_1) \subseteq \mathcal{R}(M, F)$. Because H_2H_1 is a basis for $\mathcal{R}(M, F)$, we have $\det H_2H_1 \mid \det T_2H_1$, from which it follows that $\det H_2 \mid \det T_2$. □

4.1. High level overview of the recursion

Our goal is to compute the Hermite basis H of an input $\mathcal{R}(S, F)$. Our divide and conquer algorithm factors $H = H_2H_1$ as in (17) and then has two parts. Part 1 constructs from $\mathcal{R}(S, F)$ a subproblem $\mathcal{R}(S_1, F_1)$ that has basis H_1. Part 2 then uses the result of part 1 to construct a subproblem $\mathcal{R}(S_2, F_2)$ that has basis H_2. From a cost point of view, the main challenge is to construct the two subproblems so that they satisfy the preconditions mentioned at the beginning of this section, that is, the inputs should be coprime and the modulus should be in Smith form.

4.1.1. Recursion part 1: Determining H_1

Recall that, expressed in terms of matrices, the Hermite basis H of $\mathcal{R}(S, F)$ is is the nonsingular Hermite form with minimal determinant that satisfies

$$HF = \mathbf{0} \bmod S. \tag{21}$$

Decompose F as

$$F = \begin{bmatrix} \bar{F} \\ A \end{bmatrix},$$

where A is the last n_2 rows of F. We can then rewrite (21) as

$$H_2 H_1 F = \begin{bmatrix} I_{n_1} & H_{12} \\ & \bar{H}_2 \end{bmatrix} \begin{bmatrix} \bar{H}_1 & \\ & I_{n_2} \end{bmatrix} \begin{bmatrix} \bar{F} \\ A \end{bmatrix} = \mathbf{0} \bmod S. \tag{22}$$

Part 1 is based on the following theorem.

Theorem 24. *H_1 is the Hermite basis of*

$$\mathcal{R}\left(\begin{bmatrix} S \\ A \end{bmatrix}, F \right). \tag{23}$$

Furthermore, if the inputs to $\mathcal{R}(S, F)$ are coprime, then the inputs for (23) are also coprime.

Proof. Theorem 23.1 states that H_1 is a basis of

$$\mathcal{R}\left(\begin{bmatrix} S \\ H_1 F \end{bmatrix}, F \right). \tag{24}$$

Since Hermite forms are canonical, H_1 is the Hermite basis of (24).

In order to show that the relations lattice (24) equals that of (23), Lemma 14.1 implies that it will suffice to show that

$$\mathcal{L}\left(\begin{bmatrix} S \\ H_1 F \end{bmatrix} \right) = \mathcal{L}\left(\begin{bmatrix} S \\ A \end{bmatrix} \right). \tag{25}$$

Replacing $H_1 F$ in the left hand side of (25) using

$$H_1 F = \begin{bmatrix} \bar{H}_1 & \\ & I_{n_2} \end{bmatrix} \begin{bmatrix} \bar{F} \\ A \end{bmatrix} = \begin{bmatrix} \bar{H}_1 \bar{F} \\ A \end{bmatrix},$$

and adding a block of n_1 zero rows to the matrix on the right hand side of (25), we obtain the equation

$$\mathcal{L}\left(\begin{bmatrix} S \\ \bar{H}_1 \bar{F} \\ A \end{bmatrix} \right) = \mathcal{L}\left(\begin{bmatrix} S \\ \mathbf{0} \\ A \end{bmatrix} \right), \tag{26}$$

which is equivalent to (25). To show (26), note that (22) gives the existence of an integer matrix Q such that

$$\begin{bmatrix} I & & \\ Q & I_{n_1} & H_{12} \\ & & I_{n_2} \end{bmatrix} \begin{bmatrix} S \\ \bar{H}_1 \bar{F} \\ A \end{bmatrix} = \begin{bmatrix} S \\ \mathbf{0} \\ A \end{bmatrix}. \tag{27}$$

Since the transformation in (27) is unimodular, it follows that (26) holds.

The "furthermore" claim in the statement of the theorem is obvious. □

Starting with the input (23), part 1 now computes

$$\mathcal{R}\left(\begin{bmatrix} S \\ A \end{bmatrix}, F\right) \to \mathcal{R}(T, F) \to \mathcal{R}(S_1, F_1). \tag{28}$$

The first transformation replaces the modulus in the initial relations lattice in (28) with its Hermite basis T, while the second transformation replaces T with its Smith form S_1. By Theorem 24, the initial relations lattice in (28) has coprime inputs, thus $\mathcal{R}(S_1, F_1)$ also has coprime inputs, and by Theorem 21, we have $\det S_1 = \det H_1$.

4.1.2. Recursion part 2: Determining H_2

The second part of our recursion is based on the following theorem. There the matrix T is the Hermite basis appearing in (28), which has already been computed in part 1.

Theorem 25. *The following hold:*

1. *H_2 is the Hermite basis of $\mathcal{R}(S, H_1F)$.*
2. *The Hermite basis of*

$$\begin{bmatrix} S \\ H_1F \end{bmatrix}$$

is equal to T.

Proof. Part 1 follows from Theorem 23.2 combined with the fact that Hermite forms are canonical. Part 2 follows as a corollary of the proof of Theorem 24, which shows (25). □

The initial step of part 2 is to compute $B = \mathbf{c}\mathrm{mod}(H_1F, S)$. Then Theorem 25.1 states that H_2 is the Hermite basis of $\mathcal{R}(S, B)$. However, S and B may not be coprime and so we will still need to do

$$\mathcal{R}(S, B) \to \mathcal{R}(K, C) \to \mathcal{R}(S_2, F_2). \tag{29}$$

The first transformation is to a $\mathcal{R}(K, C)$ having coprime inputs, and is based Theorem 20 which states that

$$\begin{bmatrix} T & & I \\ B & I & \\ S & & \end{bmatrix} \text{ has Hermite basis } \begin{bmatrix} T & & * \\ & I & C \\ & & K \end{bmatrix},$$

where T is the Hermite basis of Theorem 25.2 which has already been computed in part 1. The second transformation replaces K with its Smith form S_2. Since (K, C) are coprime so are (S_2, F_2). As was the case with part 1, having $\mathcal{R}(S_2, F_2)$ with coprime inputs means that $\det S_2 = \det H_2$.

4.1.3. Base case

The base case for the recursion occurs when two conditions hold: the inputs (S, F) have column dimension one, and the Hermite basis of $\mathcal{R}(S, F)$ is known to have at most one nontrivial column. Example 7 shows that the latter condition does not follow from the former.

Lemma 26. *Let $\mathcal{R}(sI_1, F)$ have coprime inputs with $s \in \mathbb{Z}_{>0}$ and $F = \mathbf{c}\mathrm{mod}(F, sI_1)$. If the Hermite basis H of $\mathcal{R}(sI_1, F)$ is known to be index $(k, 1)$, then H can be computed in $O(k\,\mathsf{M}(\log s) + \mathsf{B}(\log s))$ bit operations, or, more simply, by $O(n\,\mathsf{B}(\log s))$ bit operations. Here n is the row dimension of F.*

Proof. It follows from Theorem 21 that $\det H = \det sI_1 = s$. Since H is index $(k, 1)$, the matrices H and F in the identity $HF = \mathbf{0} \bmod S$ can be decomposed conformally as

$$\begin{array}{c} H \\ \left[\begin{array}{c|c|c} I_k & C & \\ \hline & s & \\ \hline & & I \end{array}\right] \end{array} \begin{array}{c} F \\ \left[\begin{array}{c} F_{1\ldots k,1} \\ \hline F_{k+1,1} \\ \hline \mathbf{0} \end{array}\right] \end{array} = \mathbf{0} \bmod s.$$

Then C is the unique vector in $[0, s)^{k\times 1}$ such that $CF_{k+1,1} = -F_{1\ldots k,1} \bmod s$. Computing C has the stated cost. □

Example 27. *Let $M \in \mathbb{Z}^{3\times 3}$ be the matrix from Example 7, with a Smith massager $(S, F) = (\left[\begin{array}{c} 24 \end{array}\right], \left[\begin{array}{ccc} 19 & 10 & 3 \end{array}\right]^T)$. Letting $n_1 = 2$ and $n_2 = 1$, Theorem 24 implies that H_1 is the Hermite basis of $\mathcal{R}(\left[\begin{array}{cc} 24 & 3 \end{array}\right]^T, F)$, which part 1 transforms to $\mathcal{R}(\left[\begin{array}{c} 3 \end{array}\right], F)$ since $\left[\begin{array}{c} 3 \end{array}\right]$ is the Hermite basis of $\left[\begin{array}{cc} 24 & 3 \end{array}\right]^T$. The Hermite basis of $\mathcal{R}(\left[\begin{array}{c} 3 \end{array}\right], F)$ is*

$$H_1 = \left[\begin{array}{ccc} 1 & 2 & \\ & 3 & \\ & & 1 \end{array}\right].$$

For the second part of the recursion we have $B = \mathbf{c}\bmod(H_1F, S) = \left[\begin{array}{ccc} 15 & 6 & 3 \end{array}\right]^T$. Theorem 25 implies that H_2 is the Hermite basis of $\mathcal{R}(S, B) = \mathcal{R}(\left[\begin{array}{c} 24 \end{array}\right], \left[\begin{array}{ccc} 15 & 6 & 3 \end{array}\right]^T)$, which does not have coprime inputs. Part 2 transforms $\mathcal{R}(S, B) \to \mathcal{R}(\left[\begin{array}{c} 8 \end{array}\right], \left[\begin{array}{ccc} 5 & 2 & 1 \end{array}\right]^T)$, with coprime inputs, and with an index $(3, 1)$ Hermite basis

$$H_2 = \left[\begin{array}{ccc} 1 & & 3 \\ & 1 & 6 \\ & & 8 \end{array}\right].$$

Combining H_1 and H_2 verifies that $H = H_2H_1$ as given earlier in Example 7.

4.2. Algorithm `HermiteBasis`

Algorithm `HermiteBasis` is shown in Figure 1. For an input (S, F, k, m), the precondition of the algorithm requires that the Hermite basis H of $\mathcal{R}(S, F)$ is index (k, m). For any partition $m = m_1 + m_2$ we can factor $H = H_2H_1$ into an index (k, m_1) Hermite form H_1 and an index $(k + m_1, m_2)$ Hermite form H_2 as follows:

$$\overbrace{\left[\begin{array}{cccc} I_k & * & * & \\ & \bar{H}_1 & * & \\ & & \bar{H}_2 & \\ & & & I_{n-k-m} \end{array}\right]}^{H} = \overbrace{\left[\begin{array}{ccc} I_k & & * \\ & I_{m_1} & * \\ & & \bar{H}_2 \\ & & & I_{n-k-m} \end{array}\right]}^{H_2} \overbrace{\left[\begin{array}{cccc} I_k & * & & \\ & \bar{H}_1 & & \\ & & I_{m_2} & \\ & & & I_{n-k-m} \end{array}\right]}^{H_1} \tag{30}$$

$$. \tag{31}$$

Note that the partition $m = m_1 + m_2$ in (31) corresponds to the partition $n = n_1 + n_2$ in (17), with $n_1 = k + m_1$ and $n_2 = n - (k + m_1)$.

In addition to the dimension parameter m, which the algorithm reduces by a factor of two for the subproblems, the cost also depends on the precision parameter $\det S$. As explained in the previous subsection, the algorithm ensures that the inputs to each subproblem are coprime. Although not guaranteeing an even split of the precision, the fact that the inputs are coprime does achieve that $\det S = (\det S_1)(\det S_2)$.

`HermiteBasis`$[\epsilon](S, F, k, m)$
Input:

- $S \in \mathbb{Z}^{m \times m}$, nonsingular and in Smith form.
- $F \in \mathbb{Z}^{n \times m}$, reduced column-modulo S.
- Nonnegative k with $0 \le k \le n - m$.
- ϵ, a nonzero probability

  # Precondition: Inputs to $\mathcal{R}(S, F)$ are coprime.
  # Precondition: $\mathcal{R}(S, F)$ has an index (k, m) Hermite basis.

Output:

- The Hermite basis H of $\mathcal{R}(S, F)$, or FAIL.

# Note: If any call to `SmithMassager` returns FAIL, then return FAIL.
if $m = 1$ **then**
  return the Hermite basis of $\mathcal{R}(S, F)$ *# Lemma 26*
else
  $m_1, m_2 := \lfloor m/2 \rfloor, \lceil m/2 \rceil$

  # Part 1: Compute H_1
  Let A be the last $n - k - m_1$ rows of F.
  $T :=$ the Hermite basis of $\mathcal{L}(A) + \mathcal{L}(S)$ *# Theorem 33.1*
  $V_1, S_1 :=$ `SmithMassager`$[\frac{\epsilon}{4}](T)$ *# Theorem 8*
  $F_1 := \mathbf{c}\mathrm{mod}(FV_1, S_1)$ *# Theorem 40*
  Decompose $S_1 = \mathrm{diag}(I_{m-m_1}, \bar{S}_1)$ and $F_1 = \left[\begin{array}{c|c} \mathbf{0} & \bar{F}_1 \end{array}\right]$ conformally.
  $H_1 :=$ `HermiteBasis`$[\frac{\epsilon}{4}](\bar{S}_1, \bar{F}_1, k, m_1)$

  # Part 2: Compute H_2
  $B := \mathbf{c}\mathrm{mod}(H_1 F, S)$ *# Corollary 42*
  $\begin{bmatrix} T & & * \\ & I & C \\ & & K \end{bmatrix} :=$ the Hermite basis of $\begin{bmatrix} T & & I \\ B & I & \\ S & & \end{bmatrix}$ *# Theorem 33.2*
  $V_2, S_2 :=$ `SmithMassager`$[\frac{\epsilon}{4}](K)$ *# Theorem 8*
  $F_2 := \mathbf{c}\mathrm{mod}(CV_2, S_2)$ *# Theorem 40*
  Decompose $S_2 = \mathrm{diag}(I_{m-m_2}, \bar{S}_2)$ and $F_2 = \left[\begin{array}{c|c} \mathbf{0} & \bar{F}_2 \end{array}\right]$ conformally.
  $H_2 :=$ `HermiteBasis`$[\frac{\epsilon}{4}](\bar{S}_2, \bar{F}_2, k + m_1, m_2)$

  return $H_2 H_1$

Figure 1: Algorithm `HermiteBasis`

4.3. Correctness of Algorithm `HermiteBasis`

We split the proof into two parts. Our first theorem proves correctness as per the definition of an algorithm, that is, that for any input the return value of the algorithm matches the output specification. The second theorem bounds by ϵ the probability of FAIL being returned. For these results, we implicitly assume that the algorithm is called with a sequence $[\epsilon](S, F, k, m)$ that satisfies all the specified input conditions.

Theorem 28. *Algorithm* `HermiteBasis` *is correct. That is, it either returns the Hermite basis of* $\mathcal{R}(S, F)$*, or reports* FAIL*.*

Proof. The return value in the nonrecursive "**if**" case of the algorithm is correct by definition. If any of the calls to `SmithMassager` returns FAIL, then the algorithm correctly returns FAIL. Assume henceforth that we are in the recursive "**else**" case, and that none of the calls to `SmithMassager` return FAIL. As shown in (31), we decompose the Hermite basis H of $\mathcal{R}(S, F)$ uniquely as $H = H_2 H_1$, where H_1 is index (k, m_1) and H_2 is index $(k + m_1, m_2)$. In the next two paragraphs, we will show that the inputs $(\bar{S}_1, \bar{F}_1, k, m_1)$ and $(\bar{S}_2, \bar{F}_2, k + m_1, m_2)$ that are constructed in the "**else**" case satisfy the preconditions of the algorithm, and, moreover, that H_1 is the Hermite basis of $\mathcal{R}(\bar{S}_1, \bar{F}_1)$ and H_2 is the Hermite basis of $\mathcal{R}(\bar{S}_2, \bar{F}_2)$. Correctness of the algorithm will then follow from strong induction on m, with base case $m = 1$.

Consider the construction of the first subproblem $(\bar{S}_1, \bar{F}_1, k, m_1)$. We claim that

$$\begin{aligned} \mathcal{L}(H_1) &= \mathcal{R}\left(\begin{bmatrix} S \\ A \end{bmatrix}, F\right) && (32)\\ &= \mathcal{R}(T, F) && (33)\\ &= \mathcal{R}(S_1, F_1) && (34)\\ &= \mathcal{R}(\bar{S}_1, \bar{F}_1). && (35) \end{aligned}$$

Line (32) follows from Theorem 24, while line (33) follows from Lemma 15 (transformation to Hermite modulus). Line (34) follows from Lemma 16 (transformation to Smith modulus). By Theorem 24, the input on the right hand side of (32) is coprime. This allows us to apply Theorem 21, which states that, up to trivial invariant factors, S_1 is the Smith form of H_1. By Corollary 22, the fact that H_1 is index (k, m_1) means that it has at most m_1 invariant factors. This shows that that S_1 and F_1 can be decomposed as $\mathrm{diag}(I_{m-m_1}, \bar{S}_1)$ and $F_1 = \left[\begin{array}{c|c} \mathbf{0} & \bar{F}_1 \end{array}\right]$. Line (35) now follows from Lemma 18. By construction, $\bar{S}_1 \in \mathbb{Z}^{m_1 \times m_1}$ is nonsingular and in Smith form, $\bar{F}_1 \in \mathbb{Z}^{n \times m_1}$ is reduced column-modulo $\bar{S}_1$, and the inputs to $\mathcal{R}(\bar{S}_1, \bar{F}_1)$ are coprime. Finally, from (35) we have that $\mathcal{R}(\bar{S}_1, \bar{F}_1)$ has an index (k, m_1) Hermite basis, namely H_1.

Now consider the construction of the second subproblem $(\bar{S}_1, \bar{F}_1, k + m_1, m_2)$. We claim that

$$\begin{aligned} \mathcal{L}(H_2) &= \mathcal{R}(S, H_1 F) && (36)\\ &= \mathcal{R}(S, B) && (37)\\ &= \mathcal{R}(K, C) && (38)\\ &= \mathcal{R}(S_2, F_2) && (39)\\ &= \mathcal{R}(\bar{S}_2, \bar{F}_2) && (40) \end{aligned}$$

Line (36) follows from Theorem 25. Since $B = \mathbf{c}\mathrm{mod}(H_1 F, S)$, line (37) follows from Lemma 14.2. Line (38) follows from Theorem 20, which also states that the inputs to $\mathcal{R}(K, C)$ are coprime. Correctness of lines (39) and (40), and the remainder of the proof of correctness of this subproblem construction, is analogous to that for the first subproblem given above. □

Theorem 29. *Algorithm* `HermiteBasis` *returns* FAIL *with probability at most* ϵ*.*

Proof. We use strong induction on m, with base case $m = 1$. Let $T[\epsilon](m)$ be an upper bound on the probability that the algorithm returns FAIL with a calling sequence $[\epsilon](*, *, *, m)$. Then $T[\epsilon](1) = 0$. The inductive hypothesis is $T[\epsilon](\bar{m}) \leq \epsilon$ for all $1 \leq \bar{m} < m$, some $m > 1$. We then obtain $T[\epsilon](m) \leq 2(\epsilon/4) + T[\epsilon/4](\lfloor m/2 \rfloor) + T[\epsilon/4](\lceil m/2 \rceil) \leq \epsilon$ by using the inductive hypothesis twice. □

5. Modular computation of Hermite forms

Algorithm `HermiteBasis` requires computing the Hermite bases of matrices

$$M_1 := \begin{bmatrix} A \\ S \end{bmatrix} \quad \text{and} \quad M_2 := \begin{bmatrix} T & & I \\ B & I & \\ S & & \end{bmatrix}, \tag{41}$$

where $S \in \mathbb{Z}^{m\times m}$ is a nonsingular Smith form. Part 1 of of `HermiteBasis` computes the Hermite basis T of M_1. Part 2 then constructs B to complete the definition of the matrix M_2. A salient feature of M_2 is that T is the Hermite basis of $\mathcal{L}(B) + \mathcal{L}(S)$.

As pointed out in Example 11, the Hermite bases of M_1 and M_2 can be computed using modular arithmetic over $\mathbb{Z}/(s^2)$ where s is the largest invariant factor in S. In this section we show that unimodular transformation matrices bringing M_1 and M_2 into Hermite form can also be computed over $\mathbb{Z}/(s^2)$. These transformation matrices are used in the fast algorithm develped in Section 6.

Our approach is to develop a unified algorithm that can be applied to both input matrices in (41). To this end, we augment M_1 with a square block of zeros and then transform it to Hermite form in two steps:

$$\begin{bmatrix} \mathbf{0} \\ A \\ S \end{bmatrix} \xrightarrow{1} \begin{bmatrix} T \\ A \\ S \end{bmatrix} \xrightarrow{2} \begin{bmatrix} T \\ \\ \end{bmatrix}. \tag{42}$$

Once T is computed, step 2, with B replacing A, is used to compute the Hermite form of M_2. Because of the structure of M_2, the transformation matrix achieving step 2 in (42) will reveal the blocks of the Hermite form of M_2 that we need.

We remark that, as usual, $*$ is used a placeholder for a scalar/matrix that exists but does not necessarily need to be computed explicitly. As an aid to the reader we do however identify the blocks $*_1$, $*_2$, $*_3$ and $*_4$ as these arise in future constructions. Our first result gives step 1 in (42).

Theorem 30. *Let $S \in \mathbb{Z}^{m\times m}$ be a nonsingular Smith form with largest invariant factor s, and let $A \in \mathbb{Z}^{n\times m}$ have entries reduced modulo s. Then the Hermite basis T of $\mathcal{L}(A)+\mathcal{L}(S)$ together with a matrix $E \in [0,s)^{m\times n}$ such that*

$$\left[\begin{array}{c|c|c} I & E & *_1 \\ \hline & I & \\ \hline & & I \end{array}\right] \left[\begin{array}{c} \mathbf{0} \\ \hline A \\ \hline S \end{array}\right] = \left[\begin{array}{c} T \\ \hline A \\ \hline S \end{array}\right] \tag{43}$$

for some $*_1 \in \mathbb{Z}^{m\times m}$, can be computed in $O((n+m)m^{\omega-1}\,\mathsf{B}(\log s))$ bit operations.

Proof. As pointed out in Example 11, $sI \subseteq \mathcal{L}(S)$ is in the lattice generated by the rows of (43). As such we can use Theorem 10 to compute T in the allotted time. It remains to compute E. The proof of Theorem 10 used the Howell form algorithm of [36] to compute T over $\mathbb{Z}/(s^2)$. Making use of [36, Definition 2 and Corollary 1], we observe that if we apply the Howell form algorithm to the input matrix (43) with the blocks A and S switched it will compute as a side effect two $(n+m)\times(n+m)$ matrices U and C over $\mathbb{Z}/(s^2)$ that can be written conformally as, and satisfy,

$$UC = \left[\begin{array}{c|c} \bar{U} & \\ \hline & I_n \end{array}\right] \left[\begin{array}{c|c} I_m & \bar{C} \\ \hline & I_n \end{array}\right] \left[\begin{array}{c} S \\ \hline A \end{array}\right] = \left[\begin{array}{c} T \\ \hline A \end{array}\right] \bmod s^2. \tag{44}$$

It follows from (44) that $\bar{U}S + \bar{U}\bar{C}A = T \bmod s^2$. Since $sI \subseteq \mathcal{L}(S)$, we can take $E := \mathrm{mod}(\bar{U}\bar{C}, s)$. Since $\bar{U}$ is $m\times m$ and $\bar{C}$ is $m\times n$, E can be computed in the allotted time. □

Next we show how to construct a unimodular transformation to achieve step 2 in (42):

$$\begin{bmatrix} I & & \\ C & I & *_3 \\ K & & *_4 \end{bmatrix} \begin{bmatrix} T \\ A \\ S \end{bmatrix} = \begin{bmatrix} T \\ \\ \end{bmatrix}.$$

We will actually compute a Hermite basis for a slightly more general input, one whose additional components will be of use in a later computation.

Lemma 31. *Let $S \in \mathbb{Z}^{m\times m}$ be a nonsingular Smith form, and let $T \in \mathbb{Z}^{m\times m}$ be a Hermite basis such that $S, A \subseteq \mathcal{L}(T)$. Then*

$$M = \begin{bmatrix} I & F & & \\ & T & & I \\ & A & I & \\ & S & & \end{bmatrix} \text{ has Hermite form with shape } H = \begin{bmatrix} I & G & & Q \\ & T & & * \\ & & I & C \\ & & & K \end{bmatrix}, \tag{45}$$

for new blocks G, Q, C and K. Moreover

$$\begin{bmatrix} I & C \\ & K \end{bmatrix} \text{ is the Hermite form of } \begin{bmatrix} I & -AT^{-1} \\ & -ST^{-1} \end{bmatrix} \tag{46}$$

and

$$\bar{U}M = \begin{bmatrix} I & Q & & *_2 \\ & I & & \\ & C & I & *_3 \\ & K & & *_4 \end{bmatrix} \begin{bmatrix} I & F & & \\ & T & & I \\ & A & I & \\ & S & & \end{bmatrix} = \begin{bmatrix} I & G & & Q \\ & T & & I \\ & & I & C \\ & & & K \end{bmatrix} = \bar{H}, \tag{47}$$

where $\bar{U}$ is unimodular.

Proof. We will begin by establishing the existence of a unimodular transformation $\bar{U}$ that transforms M to a matrix $\bar{H}$, with both $\bar{U}$ and $\bar{H}$ having the shape shown in (47). Along the way we will establish that (46) holds. Finally, we will do one more transformation on $\bar{H}$ to put it into Hermite form, showing that (45) holds.

Since $S, A \subseteq \mathcal{L}(T)$, the matrices ST^{-1} and AT^{-1} are integral and so we can triangularize M as

$$\begin{bmatrix} I & & & \\ & I & & \\ & -AT^{-1} & I & \\ & -ST^{-1} & & I \end{bmatrix} \begin{bmatrix} I & F & & \\ & T & & I \\ & A & I & \\ & S & & \end{bmatrix} = \begin{bmatrix} I & F & & \\ & T & & I \\ & & I & -AT^{-1} \\ & & & -ST^{-1} \end{bmatrix} \tag{48}$$

with the right hand side of (48) left equivalent to M. The transformations to Hermite form of the leading and trailing submatrices of the matrix on the right hand side of (48) have the shape

$$\begin{bmatrix} I & * \\ & I \end{bmatrix} \begin{bmatrix} I & F \\ & T \end{bmatrix} = \begin{bmatrix} I & G \\ & T \end{bmatrix} \tag{49}$$

and

$$\begin{bmatrix} I & * \\ & * \end{bmatrix} \begin{bmatrix} I & -AT^{-1} \\ & -ST^{-1} \end{bmatrix} = \begin{bmatrix} I & C \\ & K \end{bmatrix}. \tag{50}$$

Applying the transformations (49) and (50) to equation (48) we obtain

$$\begin{bmatrix} I & * & & \\ & I & & \\ & C & I & * \\ & K & & * \end{bmatrix} \begin{bmatrix} I & F & & \\ & T & & I \\ & A & I & \\ & S & & \end{bmatrix} = \begin{bmatrix} I & G & & * \\ & T & & I \\ & & I & C \\ & & & K \end{bmatrix}. \tag{51}$$

The final step to transform the right hand side of (51) to $\bar{H}$ is to add a multiple of the last block-row (containing K) to the first block-row in order to reduce entries in the top right block $*$. Performing this transformation on (51) gives equation (47). Since all the transformations performed were unimodular, so is $\bar{U}$.

Finally, notice that $\bar{H}$ in (47) may not be completely in Hermite form. In particular, if some diagonal entries of K are equal to one, then we need to add a multiple of the last block-row (containing K) to the second block-row in order to ensure all entries above the diagonals in the last block-column are reduced. Once we do this we have transformed $\bar{H}$ to its Hermite form, which has the shape of H shown in (45). □

Regarding the input matrix M of (45), the blocks T and S are always square of dimension m, but the number of rows in F and A can be arbitrary. In particular, F or A can be tall or short, and even be the $0 \times m$ matrix. In the three cases where F is empty, A is empty, and both are empty, the shape of M simplifies to

$$\begin{bmatrix} T & & I \\ A & I & \\ S & & \end{bmatrix}, \quad \begin{bmatrix} I & F & \\ & T & I \\ & S & \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} T & \\ S & I \end{bmatrix}.$$

For convenience, we will use a common bound n for the row dimension of both F and A. The dimension of M is then bounded by $2n + 2m$.

Theorem 32. *Let S, T and M be as in Lemma 31 and s be the largest invariant factor of S. If entries in A and F are reduced modulo s, and n is a common bound for the row dimension of A and F, then the the Hermite form H of M can be computed computed in $O((n+m)m^{\omega-1}\,\mathsf{B}(\log s))$ bit operations.*

Proof. Note first that the Smith form of M is $\mathrm{diag}(I, S)$, and so $sI \in \mathcal{L}(M)$. If $n = 0$ then we can directly apply Theorem 10 to compute the Hermite form of the $2m \times 2m$ matrix

$$\begin{bmatrix} T & I \\ S & \end{bmatrix}.$$

Otherwise, if A is not empty then we can decompose A as

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{bmatrix}$$

where all the blocks have m rows, except for possibly the last block A_k. Note that each

$$\begin{bmatrix} T & & I \\ A_i & I & \\ S & & \end{bmatrix} \text{ has Hermite form } \begin{bmatrix} T & & * \\ & & C_i \\ & & K \end{bmatrix}.$$

Thus the submatrix C of H can be found using at most $k \leq \lceil n/m \rceil$ Hermite form computations of matrices with dimension bounded by $3m$. This is within the allotted cost. The submatrices G and Q of the Hermite form can be found by using a similar block decomposition of F. □

6. Faster modular computation of Hermite forms

This section is devoted to establishing that the two Hermite basis computations required by Algorithm `HermiteBasis` can be done in the complexity that we require.

Theorem 33. *Let $S \in \mathbb{Z}^{m\times m}$ be a nonsingular Smith form and $A \in \mathbb{Z}^{n\times m}$ be reduced column-modulo S. The following can be computed in $O((n+m)m^{\omega-1}\,\mathsf{B}((\log\det S)/m + \log m))$ bit operations.*

1. *The Hermite basis T of $\mathcal{L}(A) + \mathcal{L}(S)$.*
2. *The blocks C and K of the Hermite form*

$$\begin{bmatrix} T & & * \\ & I & C \\ & & K \end{bmatrix} \text{ of the input matrix } \begin{bmatrix} T & & I \\ A & I & \\ S & & \end{bmatrix}.$$

Section 6.1 gives a high level overview of the algorithm supporting Theorem 33. The key computational steps are detailed in Sections 6.2 and 6.3. We defer the proof of Theorem 33 to the end of Section 6.3.

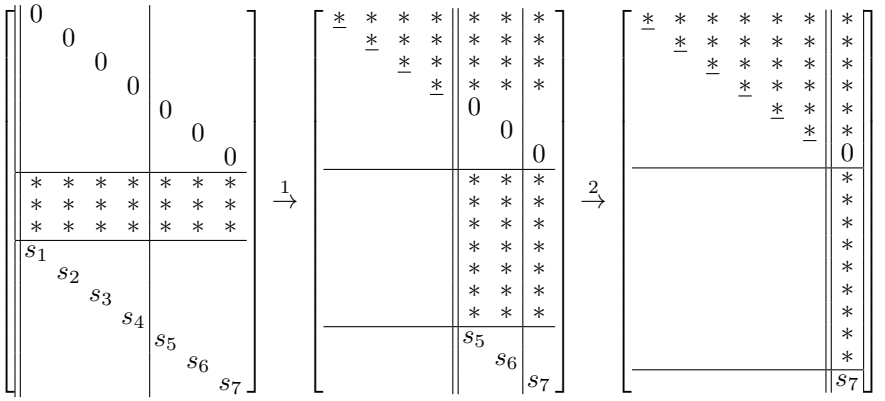
6.1. High level overview of the algorithm

Our approach is to initialize a work matrix to be equal to

$$\left[\begin{array}{c} \mathbf{0} \\ \hline A \\ \hline S \end{array}\right] \in \mathbb{Z}^{(2m+n)\times m} \tag{52}$$

and then apply a sequence of unimodular row transformations to transform it to its Hermite form in stages for $k = 1, 2, \ldots, \lceil \log_2(m+1) \rceil$. Stage 1 transforms the work matrix so that the first $m/2$ columns are in Hermite form. Stage 2 extends the number of columns in Hermite form to the first $m/2 + m/4$, stage 3 to the first $m/2 + m/4 + m/8$, and so on. At the beginning of stage k, at most $m/2^{k-1}$ columns remain to be put into Hermite form.

Example 34. *Suppose our input matrices have sizes $n = 3$ and $m = 7$, that is, $A \in \mathbb{Z}^{3\times 7}$ and $S \in \mathbb{Z}^{7\times 7}$. Then the work matrix at the start, and at end of stages 1 and 2, has the following shape:*



(53)

Note that the final stage $\xrightarrow{3}$, which transforms the last column containing s_7 to correct form, is not shown in (53). Note that each stage in (53) uses both steps in (42), that is, computing a square Hermite form, eliminating the entries below it, and reducing the entries above it.

We now specify the transformations of the work matrix at the start of each stage in more detail. Let $\bar{m}$ be the number of columns remaining to be put into correct form. Partition $S = \mathrm{diag}(S_1, \bar{S})$, where $\bar{S}$ has column dimension $\bar{m}$. (For example, in (53) the second matrix has $\bar{m} = 3$ with $\bar{S} = \mathrm{diag}(s_5, s_6, s_7)$.) Then the work matrix can be written as

$$\left[\begin{array}{c|c} T_1 & \bar{F} \\ & \mathbf{0} \\ \hline & \bar{A} \\ \hline & \bar{S} \end{array}\right], \tag{54}$$

where T_1 is a Hermite basis with $m - \bar{m}$ rows, $\bar{A}$ has row dimension $n + m - \bar{m}$ and the matrices $\bar{F}$ and $\bar{A}$ are reduced column-modulo $\bar{S}$. The row dimension of $\bar{F}$ and $\bar{A}$ are thus bounded by m and $n + m$, respectively, independent of the particular stage. At the start of stage 1, $\bar{m} = m$ and (54) coincides with the input matrix (52). At the end of the last stage, $\bar{m} = 0$ and (54) is in Hermite form.

If $\bar{m} > 0$, then our goal at the current stage is to transform the first half of the last $\bar{m}$ columns of (54) to Hermite form. To this end, decompose $\bar{S} = \mathrm{diag}(\bar{S}_1, \bar{S}_2)$, where $\bar{S}_1$ has dimension $\bar{m}_1 := \lceil \bar{m}/2 \rceil$, and $\bar{S}_2$ has dimension $\bar{m}_2 := \lfloor \bar{m}/2 \rfloor$. Conformally decompose $\bar{F} = \left[\begin{array}{c|c} \bar{F}_1 & \bar{F}_2 \end{array}\right]$ and $\bar{A} = \left[\begin{array}{c|c} \bar{A}_1 & \bar{A}_2 \end{array}\right]$. Then we can

refine the decomposition of the work matrix in (54) as

$$\left[\begin{array}{c|c||c|c} T_1 & & \bar{F}_1 & \bar{F}_2 \\ & & \mathbf{0} & \\ & & & \mathbf{0} \\ \hline & & \bar{A}_1 & \bar{A}_2 \\ \hline & & \bar{S}_1 & \\ & & & \bar{S}_2 \end{array}\right]. \tag{55}$$

The current stage now applies a pair of unimodular transformations to (55):

$$\left[\begin{array}{cc|c|c} I & Q & & *_2 \\ & I & & \\ & & I & \\ \hline C & & I & *_3 \\ \hline K & & & *_4 \\ & & & & I \end{array}\right] \left[\begin{array}{cc|c|c} I & & & \\ & I & E & *_1 \\ & & I & \\ \hline & & I & \\ \hline & & & I \\ & & & & I \end{array}\right] \left[\begin{array}{c|c||c|c} T_1 & & \bar{F}_1 & \bar{F}_2 \\ & & \mathbf{0} & \\ & & & \mathbf{0} \\ \hline & & \bar{A}_1 & \bar{A}_2 \\ \hline & & \bar{S}_1 & \\ & & & \bar{S}_2 \end{array}\right] = \left[\begin{array}{c|c||c|c} T_1 & & \bar{G}_1 & * \\ & & \bar{T}_1 & * \\ & & & \mathbf{0} \\ \hline & & & * \\ \hline & & & * \\ & & & \bar{S}_2 \end{array}\right]. \tag{56}$$

The right hand side of (56) has the shape required for the next stage. We remark that the blocks $*_1$, $*_2$, $*_3$ and $*_4$ encode unimodular row operations used to keep entries reduced modulo the diagonal entries of $\bar{S}_1$. We will need to show the existence of these blocks, but do not need to compute them explicitly since the blocks above $\bar{S}_2$ in the transformed matrix do not depend on them.

Before detailing the construction of the transformation, and the computation of the right hand side of (56), we explain how this incremental approach of putting only half of the remaining columns into correct form allows us to obtain an overall cost estimate that depends on the average bitlength $(\log \det S)/m$ of the invariant factors of S. We exploit a

$$\text{dimension} \;\times\; \text{precision} \;\leq 2(\log \det S) \tag{57}$$

invariant that holds for each stage. The "dimension" in (57) refers to the column dimension $\bar{m}$ of $\bar{S} = \operatorname{diag}(\bar{S}_1, \bar{S}_2)$, and "precision" refers to the bitlength of the largest invariant factor s of $\bar{S}_1$. In particular, the needed blocks of the unimodular transformation in (56) will be computed modulo s.

Lemma 35. *Let $S \in \mathbb{Z}^{m\times m}$ be a nonsingular Smith form. Partition*

$$\begin{aligned} S &= \operatorname{diag}(S_1, \bar{S}) \\ &= \operatorname{diag}(S_1, \operatorname{diag}(\bar{S}_1, \bar{S}_2)) \end{aligned}$$

with the dimension $\bar{m}$ of $\bar{S}$ arbitrary, but the second partition splitting $\bar{S}$ evenly, with $\bar{S}_1$ of dimension $\lceil \bar{m}/2 \rceil$. Let s be the largest invariant factor of $\bar{S}_1$. Then $\log s \leq 2(\log \det S)/\bar{m}$.

Proof. $\bar{S}_2$ has dimension $\bar{m} - \lceil \bar{m}/2 \rceil = \lfloor \bar{m}/2 \rfloor$. Since s is in $\bar{S}_1$, and s divides all invariant factors in $\bar{S}_2$, there are at least $\lfloor \bar{m}/2 \rfloor + 1 \geq \bar{m}/2$ invariant factors in $\bar{S}$ that are multiples of s. Since $\det S$ is the product of all invariant factors, we have $s^{\bar{m}/2} \leq \det S$. Solving for $\log s$ gives the result. □

6.2. Constructing the unimodular transformations

We now show how to construct the unimodular transformations in (56). The transformations depend only on the columns containing $\bar{S}_1$. Also, note that the $\bar{m}_2$ rows containing $\bar{S}_2$, and the $\bar{m}_2$ rows containing the block $\mathbf{0}$ directly above $\bar{A}_2$, are not modified, so we can ignore these rows for now. The transformation

we need to compute is U_2U_1 satisfying

$$\overset{U_2}{\left[\begin{array}{cc|c|c} I & Q & & *_2 \\ & I & & \\ \hline & C & I & *_3 \\ \hline & K & & *_4 \end{array}\right]} \overset{U_1}{\left[\begin{array}{c|cc|c} I & & & \\ & I & E & *_1 \\ \hline & & I & \\ \hline & & & I \end{array}\right]} \left[\begin{array}{c} \bar{F}_1 \\ \mathbf{0} \\ \hline \bar{A}_1 \\ \hline \bar{S}_1 \end{array}\right] = \left[\begin{array}{c} \bar{G}_1 \\ \bar{T}_1 \\ \hline \\ \hline \end{array}\right], \tag{58}$$

where the input blocks $\bar{F}_1$ and $\bar{A}_1$ are reduced column-modulo $\bar{S}_1$, and the output blocks in the right hand side are such that

$$\left[\begin{array}{cc} I & \bar{G}_1 \\ & \bar{T}_1 \end{array}\right]$$

is in Hermite form. Recall that $\bar{m} = \bar{m}_1 + \bar{m}_2$. Comparing with (54), we see that $\bar{F}_1$, $\bar{A}_1$ and $\bar{S}_1$ have row dimension equal to $m - \bar{m}$, $n + m - \bar{m}$ and $\bar{m}_1$, respectively. Instead of these exact dimensions, for cost estimates we will uses the upper bounds m, $n + m$ and $\bar{m}$, respectively. More generally, at any stage, the dimensions of the diagonal blocks of U_1 and U_2 are bounded by the dimension of the corresponding diagonal blocks in $\mathrm{diag}(I_m, I_{\bar{m}}, I_{n+m}, I_{\bar{m}})$. For example, E has dimension bounded by $\bar{m} \times (n + m)$. By using these upper bounds, the only parameters that are specialized to the current stage for the purpose of the cost analysis are $\log s$, where s is the largest invariant factor of $\bar{S}_1$, and $\bar{m}$.

Observe that the first transformation

$$\overset{U_1}{\left[\begin{array}{c|cc|c} I & & & \\ & I & E & *_1 \\ \hline & & I & \\ \hline & & & I \end{array}\right]} \left[\begin{array}{c} \bar{F}_1 \\ \mathbf{0} \\ \hline \bar{A}_1 \\ \hline \bar{S}_1 \end{array}\right] = \left[\begin{array}{c} \bar{F}_1 \\ \hline \bar{T}_1 \\ \hline \bar{A}_1 \\ \hline \bar{S}_1 \end{array}\right] \tag{59}$$

simply computes the Hermite form $\bar{T}_1$ along with the matrix E which puts the matrix into this form. The second transformation U_2 then continues to transform the right hand side of (59) to the right hand side of (58).

Lemma 36. *There exists an algorithm that computes matrices E, Q, C and K creating unimodular matrices U_1 and U_2 satisfying (58) with cost $O((n+m)\bar{m}^{\omega-1}\,\mathsf{B}(\log s))$ bit operations, where s is the largest invariant factor of $\bar{S}_1$. The algorithm computes in the same time the blocks $\bar{G}_1$ and $\bar{T}_1$ on the right hand side of (58). All the computed blocks have entries from $[0, s]$.*

Proof. Theorem 30 gives the result for computing E and $\bar{T}_1$. By Lemma 31,

$$\left[\begin{array}{cc|c} I & \bar{F}_1 & \\ & \bar{T}_1 & I \\ & \bar{A}_1 & I \\ & \bar{S}_1 & \end{array}\right] \text{ has Hermite form with shape } \left[\begin{array}{cc|c} I & \bar{G}_1 & Q \\ & \bar{T}_1 & * \\ \hline & & I \quad C \\ & & K \end{array}\right], \tag{60}$$

with the blocks $\bar{G}_1$, Q, C and K satisfying the requirements in the statement of the lemma. By Theorem 32 the Hermite form in (60) can be computed in the allotted time. □

6.3. Applying the unimodular transformations

In order to complete the current stage, we need to find the last $\bar{m}_2$ columns of the matrix on the right of (56), which are given by

$$\overset{U_2}{\left[\begin{array}{cc|c|c} I & Q & & *_2 \\ & I & & \\ \hline & C & I & *_3 \\ \hline & K & & *_4 \end{array}\right]} \overset{U_1}{\left[\begin{array}{c|cc|c} I & & & \\ & I & E & *_1 \\ \hline & & I & \\ \hline & & & I \end{array}\right]} \left[\begin{array}{c} \bar{F}_2 \\ \hline \\ \hline \bar{A}_2 \\ \hline \end{array}\right] = \left[\begin{array}{c} \bar{F}_2 + QE\bar{A}_2 \\ \hline E\bar{A}_2 \\ \hline \bar{A}_2 + CE\bar{A}_2 \\ \hline KE\bar{A}_2 \end{array}\right]. \tag{61}$$

Considering that the diagonal elements in the block $\bar{S}_2$ in (56) can be used to arbitrarily reduce entries in the blocks above it (operations which correspond to unimodular row transformations), it will suffice to compute the right hand side of (61) column-modulo $\bar{S}_2$.

Lemma 37. *Given the blocks E, Q, C and K from Lemma 36, the matrix on the right of (61) can be computed column-modulo $\bar{S}_2$ in $O((n+m)\bar{m}^{\omega-1}\mathsf{M}((\log\det S)/\bar{m}+\log\bar{m}))$ bit operations.*

Proof. The right hand side of (61) is equal to

$$\left[\begin{array}{c} \bar{F}_2 \\ \hline \bar{A}_2 \\ \hline \end{array}\right] + \left[\begin{array}{c} Q \\ \hline I \\ \hline C \\ \hline K \end{array}\right] E\bar{A}_2. \tag{62}$$

We will compute (62) column-modulo $\bar{S}_2$ in three steps. We remark that, for readability, we have gathered together all of the results that rely on partial linearization in the next Section 7. As such, we make forward references to Theorems 40 and 43 that establish the cost bounds for the matrix multiplications column-modulo $\bar{S}_2$ that we require here in steps 1 and 2.

Step 1: Compute $B = \mathbf{c}\mathrm{mod}(E\bar{A}_2, \bar{S}_2)$. The row dimension of E and column dimension of $\bar{A}_2$ are bounded by $\bar{m}$, and the inner dimension of the product $E\bar{A}_2$ is bounded by $n+m$. Since we want the result of $E\bar{A}_2$ column-modulo $\bar{S}_2$, we will appeal to Theorem 43 to do the product. In particular, from Lemma 36 we have that $E = \mathbf{r}\mathrm{mod}(E, 2sI)$, and, moreover, since E has at most $\bar{m}$ rows, we have the bound

$$\begin{aligned} \log\det 2sI &\leq \bar{m}(\log 2s) \\ &= \bar{m}\log 2 + \bar{m}\log s \\ &\leq \bar{m}\log 2 + 2\log\det S, \end{aligned} \tag{63}$$

where (63) follows from Lemma 35. Let D be equal to the right hand side of (63). By assumption, $\bar{A}_2 = \mathbf{c}\mathrm{mod}(\bar{A}_2, \bar{S}_2)$, and, moreover, since $\bar{S}_2$ is a submatrix of S, we have the bound $\log\det\bar{S}_2 \leq \log\det S \leq D$. It now follows from Theorem 43 that B can be computed in the allotted time.

Step 2: Compute

$$\mathbf{c}\mathrm{mod}\left(\left[\begin{array}{c} Q \\ \hline I \\ \hline C \\ \hline K \end{array}\right] B, \bar{S}_2\right).$$

Both matrices in the product have column dimension bounded by $\bar{m}$. By Lemma 36, the first matrix is reduced column modulo $2sI$, and by construction in step 2, the second matrix B is reduced column modulo $\bar{S}_2$. By Theorem 40, the product can be done in the allotted time with D equal to the right hand side of (63).

Step 3: Finally, add the result of step 2 to the matrix containing $\bar{F}_2$ in (62), and reduce the result column modulo $\bar{S}_2$. This can be done in the allotted time. □

Proof of Theorem 33. We first give the algorithm to transform the matrix in (52) to its Hermite form. To start stage 1, initialize the work matrix to be (52), with all $\bar{m} = m$ columns remaining to be transformed to correct form. In general, the work matrix has the shape shown in (54), with $\bar{S}$ of dimension $\bar{m}$. Decompose the work matrix as in (55), with $\bar{S}_2$ of dimension $\lfloor\bar{m}/2\rfloor$, and use Lemma 36 and 37 to compute a unimodular transformation of the work matrix so it has the shape shown on the right of (56). This completes one stage. Each stage puts the next $\lceil\bar{m}/2\rceil$ columns into correct form. Because each stage is a unimodular row transformation (over $\mathbb{Z}$) the resulting matrix at the end of all stages is the Hermite form of the starting matrix (52).

We now bound the cost of a particular stage in terms of $\bar{m}$. By Lemma 35, the $\log s$ term in the cost estimate of Lemma 36 can be bounded from above by $2(\log\det S)/\bar{m}$. By Lemmas 36 and 37, the cost of one stage is then

$$O((n+m)\bar{m}^{\omega-1}\,\mathsf{B}((\log\det S)/\bar{m}+\log\bar{m})) \tag{64}$$

bit operations. Let $D = \log\det S + m\log m$. Then

$$(\log\det S)/\bar{m}+\log\bar{m} \le D/\bar{m} \tag{65}$$

and

$$\begin{aligned}\bar{m}^{\omega-1}\,\mathsf{B}(D/\bar{m}) &= m^{\omega-1}(\bar{m}/m)^{\omega-1}\,\mathsf{B}((m/\bar{m})(D/m))\\ &\le m^{\omega-1}(\bar{m}/m)^{\omega-1}\,\mathsf{B}(m/\bar{m})\mathsf{B}(D/m)\\ &\in O(m^{\omega-1}\mathsf{B}(D/m))\times(\bar{m}/m)^{\epsilon_0},\end{aligned} \tag{66}$$

where (66) used $\mathsf{B}(m/\bar{m}) \in O((m/\bar{m})^{\omega-1-\epsilon_0})$. Each stage decreases $\bar{m}$ by at least half, so the sum of $(\bar{m}/m)^{\epsilon_0}$ over all stages is a constant. Substituting (65) and (66) into (64) gives the result.

Now consider part 2 of the theorem. Consider running a simplified version of the algorithm just described when T is already known: the input matrix is now as in (52) but with the block $\mathbf{0}$ replaced by T. In this case equation (58) changes as follows: the computation of U_1 is entirely avoided (ie, U_1 is the identity) and the blocks Q and $*_2$ in U_2 are zero since $\bar{F}_1 = \bar{G}_1$.

Considering the shape of U_2, if we apply all the unimodular transformations computed in the simplified algorithm just described to the input matrix

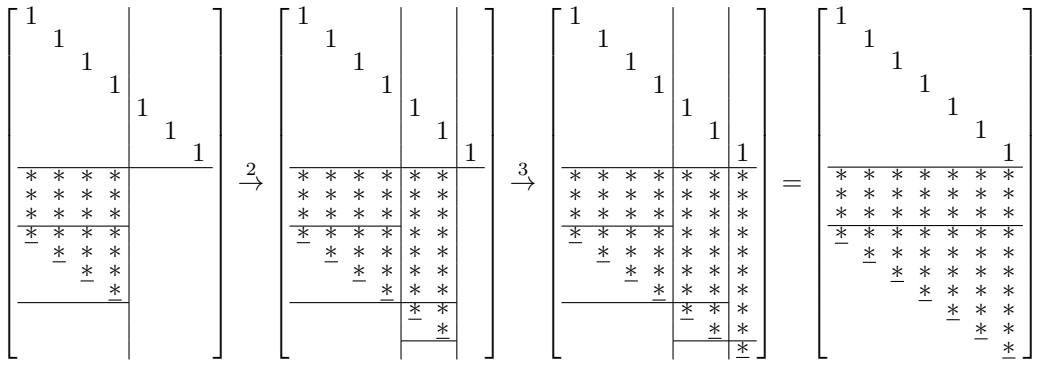
$$\begin{bmatrix} T & & I_m \\ A & I_n & \\ S & & \end{bmatrix} \tag{67}$$

we will transform it (via unimodular row transformations over $\mathbb{Z}$) to the matrix

$$\begin{bmatrix} T & & I_m \\ & I_n & C \\ & & K \end{bmatrix} \tag{68}$$

which has trailing $(n+m)\times(n+m)$ submatrix in Hermite form. □

Example 38. *As in Example 34, suppose our input matrices have sizes $n = 3$ and $m = 7$, that is, $A \in \mathbb{Z}^{3\times 7}$ and $S \in \mathbb{Z}^{7\times 7}$. But now we also have the Hermite basis $T \in \mathbb{Z}^{n\times n}$. We illustrate the transform of (67) to (68), focusing on the last block-column of (67). The process becomes*



Stage 1 starts by filling in the first four columns of C and K below the first four diagonals. Stage 2 deposits the next two columns of C and K below the fourth and fifth diagonals, leaving the first four columns unchanged. Stage 3 fills in the last column and completes.

Example 39. *Let* $A = \begin{bmatrix} 1 & 5 & 19 \end{bmatrix}$ *and* $S = \mathrm{diag}(2, 6, 72)$. *Then*

$$T = \begin{bmatrix} 1 & 1 & 5 \\ & 2 & 4 \\ & & 6 \end{bmatrix} \quad \textit{is the Hermite form of} \quad \begin{bmatrix} A \\ S \end{bmatrix}.$$

We will show how the algorithm transforms

$$\left[\begin{array}{c|c} T & I_3 \\ A & \\ S & \end{array}\right] \xrightarrow{1} \cdot \xrightarrow{2} \left[\begin{array}{c|c} T & I_3 \\ & C \\ & K \end{array}\right]$$

to produce C *and* K. *Transforming the first two columns in stage 1 and then the last column in stage 2, we obtain*

$$\left[\begin{array}{ccc||ccc} 1 & 1 & 5 & 1 & & \\ & 2 & 4 & & 1 & \\ & & 6 & & & 1 \\ \hline 1 & 5 & 19 & & & \\ \hline 2 & & & & & \\ & 6 & & & & \\ & & 72 & & & \end{array}\right] \xrightarrow{1} \left[\begin{array}{ccc||ccc} 1 & 1 & 5 & 1 & & \\ & 2 & 4 & & 1 & \\ & & 6 & & & 1 \\ \hline & & 24 & 1 & 0 & \\ & & 18 & 2 & 2 & \\ & & 18 & & 3 & \\ \hline & & 72 & & & \end{array}\right] \xrightarrow{2} \left[\begin{array}{ccc||ccc} 1 & 1 & 5 & 1 & & \\ & 2 & 4 & & 1 & \\ & & 6 & & & 1 \\ \hline & & & 1 & 0 & 8 \\ \hline & & & 2 & 2 & 9 \\ & & & & 3 & 10 \\ & & & & & 12 \end{array}\right]$$

giving

$$C := \begin{bmatrix} 1 & 0 & 8 \end{bmatrix} \quad \textit{and} \quad K := \begin{bmatrix} 2 & 2 & 9 \\ & 3 & 10 \\ & & 12 \end{bmatrix}.$$

We remark that the first stage computes modulo 6^2 *while the second stage computes modulo* 72^2. *The computation that produces column 3 in the second matrix uses formula (61), but with* U_1 *the identity matrix, and blocks* Q *and* $*_2$ *in* U_2 *the zero matrix.*

7. Matrix multiplication modulo diagonal matrices

Algorithm `HermiteBasis` makes use of matrix multiplication modulo a diagonal matrix in the operations $F_1 := \mathbf{c}\mathrm{mod}(FV_1, S_1)$, $F_2 := \mathbf{c}\mathrm{mod}(CV_2, S_2)$ and $B := \mathbf{c}\mathrm{mod}(H_1F, S)$. The first two of these have the form "tall × square," while the last is of the form "Hermite × tall." The Hermite basis computation in Section 6 requires an additional type of the form "wide × tall" for the operation $B = \mathbf{c}\mathrm{mod}(E\bar{A}_2, \bar{S}_2)$. In this section we show how to do these operations with a good bit complexity rather than algebraic complexity. Our constructions are based on partial linearization. We begin by stating the results we need. The actual construction and proofs of Theorem 40 and 43 are in Section 7.1.

Consider first computing the product of an $n \times m$ matrix A and $m \times m$ matrix B, with $n \geq m$:

$$\begin{bmatrix} A \\ \\ \end{bmatrix} \begin{bmatrix} B \end{bmatrix} = \begin{bmatrix} C \\ \\ \end{bmatrix}.$$

In an algebraic model, C is computed in $O(nm^{\omega-1})$ arithmetic operations from the domain of entries. Here we consider the cost in bit operations of computing AB over the integers, but in the following scenario where we are given nonsingular diagonal matrices E and F:

(i) We know that $A = \mathbf{c}\mathrm{mod}(A, E)$ and $B = \mathbf{c}\mathrm{mod}(B, F)$.

(ii) We only need $\mathbf{c}\mathrm{mod}(AB, F)$.

Let D be a common upper bound for both $\log\det E$ and $\log\det F$. Then D is an upper bound for the sum of the logarithms of the m diagonal entries of both E and F, and the average bitlength of the columns of A and B is bounded by $O(D/m+1)$. Up to log factors, Theorem 40 shows that $\mathbf{cmod}(AB,F)$ can be computed in a number of bit operations equal to the algebraic cost $O(nm^{\omega-1})$ times $D/m+1$.

Theorem 40. *Let $E,F\in\mathbb{Z}_{\geq 0}^{m\times m}$ be nonsingular diagonal matrices. Let $A=\mathbf{cmod}(A,E)\in\mathbb{Z}^{n\times m}$ and $B=\mathbf{cmod}(B,F)\in\mathbb{Z}^{m\times m}$, with $n\geq m$. Then $\mathbf{cmod}(AB,F)$ can be computed in $O(nm^{\omega-1}\,\mathsf{M}(D/m+\log m))$ bit operations, where D is an upper bound for $\log\det E$ and $\log\det F$.*

We will also need to apply the column-modulo matrix multiplication technique of Theorem 40 with an input matrix A that may not necessarily have all entries nonnegative. To this end, let $A^{(+)}$ denote the the matrix A but with all negative entries zeroed out. Then $A=A^{(+)}-(-A)^{(+)}$ with $A^{(+)}$ and $(-A)^{(+)}$ being over $\mathbb{Z}_{\geq 0}$.

Corollary 41. *Let $F\in\mathbb{Z}_{\geq 0}^{m\times m}$ be a nonsingular diagonal matrix. Let $A\in\mathbb{Z}^{n\times m}$ and $B=\mathbf{cmod}(B,F)\in\mathbb{Z}^{m\times m}$, with $n\geq m$. Then $\mathbf{cmod}(AB,F)$ can be computed in $O(nm^{\omega-1}\,\mathsf{M}(D/m+\log m))$ bit operations, where D is an upper bound for both $\log\det F$ and the sum of the bitlengths of the columns of A.*

Proof. Let d_i be the bitlength of column i of $A\in\mathbb{Z}^{n\times m}$, $1\leq i\leq m$. Then $A^{(+)}$ and $(-A)^{(+)}$ are reduced column-modulo E, where $E=\mathrm{diag}(2^{d_1},2^{d_2},\ldots,2^{d_m})$. By assumption, $D\geq\log_2\det E$. Use Theorem 40 to compute $C_1:=\mathbf{cmod}(A^{(+)}B,F)$ and $C_2:=\mathbf{cmod}((-A)^{(+)}B,F)$ in the allotted time, and then compute $\mathbf{cmod}(AB,F)$ as $\mathbf{cmod}(C_1-C_2,F)$. □

Corollary 42. *Suppose we are given as input*

1. *$H\in\mathbb{Z}^{n\times n}$, a nonsingular Hermite basis with at most m nontrivial columns,*
2. *$S\in\mathbb{Z}^{m\times m}$, a nonsingular Smith form with $m\leq n$, and*
3. *$M\in\mathbb{Z}^{n\times m}$, satisfying $M=\mathbf{cmod}(M,S)$.*

Then $\mathbf{cmod}(HM,S)$ can be computed in $O(nm^{\omega-1}\,\mathsf{M}(D/m+\log m))$ bit operations, where D is an upper bound for $\log\det H$ and $\log\det S$.

Proof. Since $HM=(H-I)M+M$, it will suffice to show how to compute $\mathbf{cmod}((H-I)M,S)$ in the allotted time. By assumption, $H-I$ has at least $n-m$ zero columns. Let $\mathcal{S}\subseteq\{1,2,\ldots,n\}$ be a subset of column indices, with $|\mathcal{S}|=m$, which captures all the nonzero columns of $H-I$, and let $\bar{H}\in\mathbb{Z}^{n\times m}$ be the submatrix of $H-I$ comprised of columns $\mathcal{S}$. Let $\bar{M}\in\mathbb{Z}^{m\times m}$ be the subset of M comprised of rows $\mathcal{S}$. Then $\mathbf{cmod}((H-I)M,S)=\mathbf{cmod}(\bar{H}\bar{M},S)$, which by Theorem 40 can be computed in the stated time. □

Theorem 43 gives the analogous result of Theorem 40 in the case where A is transposed:

$$\begin{bmatrix} & A & \end{bmatrix}\begin{bmatrix} B \\ \\ \end{bmatrix}=\begin{bmatrix} C \end{bmatrix}.$$

Theorem 43. *Let $E,F\in\mathbb{Z}_{\geq 0}^{m\times m}$ be nonsingular diagonal matrices. Let $A=\mathbf{rmod}(A,E)\in\mathbb{Z}^{m\times n}$ and $B=\mathbf{cmod}(B,F)\in\mathbb{Z}^{n\times m}$, with $n\geq m$. Then $\mathbf{cmod}(AB,F)$ can be computed in $O(nm^{\omega-1}\,\mathsf{M}(D/m+\log m))$ bit operations, where D is upper bound for $\log\det E$ and $\log\det F$.*

7.1. Proofs of Theorems 40 and 43

Partial linearization. We will make use of partial linearization [34] in the proofs of Theorems 40 and 43 and so begin by defining some notation used for this construction. For a modulus $X\in\mathbb{Z}_{>0}$ and length $t\in\mathbb{Z}_{\geq 0}$,

define $\vec{X}^{(t)}$ to be the expansion/compression vector

$$\vec{X}^{(t)} := \begin{bmatrix} 1 \\ X \\ \vdots \\ X^{t-1} \end{bmatrix} \in \mathbb{Z}^{t\times 1}.$$

Then premultiplying $u \in \mathbb{Z}^{1\times *}$ expands one row into t rows:

$$\vec{X}^{(t)}u = \begin{bmatrix} u \\ Xu \\ \vdots \\ X^{t-1}u \end{bmatrix} \in \mathbb{Z}^{t\times *}.$$

Postmultiplying $v = \begin{bmatrix} v_0 & v_1 & \cdots & v_{t-1} \end{bmatrix} \in \mathbb{Z}^{*\times t}$ compresses t columns into one column:

$$v\vec{X}^{(t)} = \begin{bmatrix} v_0 + v_1X + \cdots + v_{t-1}X^{t-1} \end{bmatrix} \in \mathbb{Z}^{*\times 1}.$$

For a tuple $\vec{t} = (t_1, t_2, \ldots, t_m)$ of nonnegative integers, define

$$\vec{X}^{(\vec{t})} := \begin{bmatrix} \vec{X}^{(t_1)} & & & \\ & \vec{X}^{(t_2)} & & \\ & & \ddots & \\ & & & \vec{X}^{(t_m)} \end{bmatrix} \in \mathbb{Z}^{|\vec{t}|\times m},$$

where $|\vec{t}| = \sum_i t_i$.

Proof of Theorem 40. Let $X \in \mathbb{Z}_{>0}$ be the smallest power of two such that that $\log X \geq D/m$. We will construct $\bar{A} \in [0, X)^{n\times|\vec{e}|}$ with $|\vec{e}| < 2m$, and $\bar{B} \in [0, X)^{|\vec{e}|\times|\vec{f}|}$ with $|\vec{f}| < 2m$, such that

$$\mathbf{c}\mathrm{mod}(AB, F) = \mathbf{c}\mathrm{mod}(\bar{A}\bar{B}\vec{X}^{(\vec{f})}, F). \tag{70}$$

Let $\bar{A}$ be the matrix obtained from A by replacing column i $(1 \leq i \leq m)$ with the $n \times e_i$ matrix corresponding to the coefficients of its X-adic expansion. Note that we can take $e_i \geq 0$ to be minimal such that $X^{e_i} \geq E_{ii}$, giving the upper bound

$$\begin{aligned} e_i &< 1 + (\log E_{ii})/(\log X) \\ &\leq 1 + (\log E_{ii})/(D/m) \\ &= 1 + (m/D)\log E_{ii}. \end{aligned}$$

Since $D \geq \log\det E$, we see that the column dimension $|\vec{e}| = \sum_{i=1}^{m} e_i$ of $\bar{A}$ satisfies $|\vec{e}| < 2m$.

Note that $A = \bar{A}\vec{X}^{(\vec{e})}$ and thus $AB = \bar{A}\vec{X}^{(\vec{e})}B$. Define $\hat{B} = \mathbf{c}\mathrm{mod}(\vec{X}^{(\vec{e})}B, F) \in \mathbb{Z}^{|\vec{e}|\times m}$. Then $\mathbf{c}\mathrm{mod}(AB, F) = \mathbf{c}\mathrm{mod}(\bar{A}\hat{B}, F)$. Let $\bar{B}$ be the matrix obtained from $\hat{B}$ by replacing column i $(1 \leq i \leq m)$ with the $e \times f_i$ matrix corresponding to its X-adic expansion. Similar to $\bar{A}$, we can take $f_i \geq 0$ to be minimal such that $X^{f_i} \geq F_{ii}$, in which case the column dimension $|\vec{f}|$ of $\bar{B}$ satisfies $|\vec{f}| < 2m$. At this point we have constructed $\bar{A}$ and $\bar{B}$ such that (70) is satisfied.

The complete recipe to compute $C = \mathbf{c}\mathrm{mod}(AB, F)$ using partial linearization is as follows:

step 1 : Construct $\bar{A}$ such that $A = \bar{A}\vec{X}^{(\vec{e})}$	: column linearization
step 2 : Compute $\hat{B} := \mathbf{c}\mathrm{mod}(\vec{X}^{(\vec{e})}B, F)$	: row expansion
step 3 : Construct $\bar{B}$ such that $\hat{B} = \bar{B}\vec{X}^{(\vec{f})}$	: column linearization
step 4 : $\bar{C} := \bar{A}\bar{B}$	: product over $\mathbb{Z}$
step 5 : $C := \mathbf{c}\mathrm{mod}(\bar{C}\vec{X}^{(\vec{f})}, F)$	: column compression

We now give a cost analysis in terms of bit operations. The "product over $\mathbb{Z}$" step 4 dominates the cost so we consider it first. Since both $\bar{A}$ and $\bar{B}$ have column dimension $< 2m$, and $\bar{A}$ has row dimension $n \geq m$, the cost is $O(nm^{\omega-1}\,\mathsf{M}(\log X + \log m))$. (The $\log m$ additive term in $\mathsf{M}(\log X + \log m)$ accounts for the fact that entries of $\bar{C} = \bar{A}\bar{B}$ can have magnitude $O(mX)$ since they are the result of inner products of length $< 2m$.) Since $\log X \in O(D/m)$, the cost is as stated in the theorem. Next we show that steps 1, 2, 3 and 5 can be done in a time that is dominated by the cost bound just derived for step 4.

Since X is a power of 2, the two "column linearization" steps 1 and 3 can be done in linear time $O(nm \log X)$.

The "row expansion" step 2 can be done using $|\vec{e}| - m \leq m$ iterations of the following operation: multiply by X a row vector in $\mathbb{Z}^{1\times m}$ that is reduced column-modulo F, and then reduce it column-modulo F. The cost is $O(m \sum_{i=1}^{m} \mathsf{M}(\log X + \log F_{ii}))$, which simplifies to $O(m\,\mathsf{M}(m(\log X + D/m)))$ using first the superlinearity of M and then the assumption that $D \geq \log\det F$. Simplifying the $\mathsf{M}(m(\log X + D/m))$ term in this cost estimate using $\mathsf{M}(m(\log X + D/m)) \leq \mathsf{M}(m)\mathsf{M}(\log X + D/m)$ and $\mathsf{M}(m) \in O(m^{\omega-1-\epsilon_0})$ shows that the cost of step 2 is bounded by $O(m^{\omega-\epsilon_0}\,\mathsf{M}(D/m + \log X))$ bit operations.

For the "column compression" step 5 we first compute $\bar{C}\vec{X}^{(\vec{f})}$. Since $||\bar{C}|| \in O(mX)$ and X is a power of 2, the binary representation of entries in $\bar{C}\vec{X}^{(\vec{f})}$ can be computed, in order from least to most significant bit, with $n(|\vec{f}| - m) < nm$ additions of integers having bitlength $O(\log X + \log m)$, that is, in $O(nm(\log X + \log m))$ bit operations. It remains to reduce entries in $\bar{C}\vec{X}^{(\vec{f})}$ column-modulo F. A similar analysis as done for the "row expansion" step 2, now allowing for the fact that column i of $\bar{C}\vec{X}^{(\vec{f})}$ can have bitlength $O(\log F_{ii} + \log m)$, shows that the cost of this column-modulo F operation is bounded by $O(nm^{\omega-1-\epsilon_0}\,\mathsf{M}(D/m + \log X + \log m))$ bit operations. □

Proof of Theorem 43. Let $X \in \mathbb{Z}_{>0}$ be the smallest power of two such that that $\log X \geq D/m$. We are going to construct an $\bar{A} \in [0, X)^{|\vec{e}|\times n}$ with $|\vec{e}| < 2m$, and $\bar{B} \in [0, X)^{n\times|\vec{f}|}$ with $|\vec{f}| < 2m$, with

$$\mathbf{c}\mathrm{mod}(AB, F) = \mathbf{c}\mathrm{mod}\big(\mathrm{Transpose}(\vec{X}^{(\vec{e})})\bar{A}\bar{B}\vec{X}^{(\vec{f})}, F\big). \tag{71}$$

Let $\bar{A}$ be the matrix obtained from A by replacing row i ($1 \leq i \leq m$) with the $e_i \times n$ matrix corresponding to the coefficients of its X-adic expansion. Let $\bar{B}$ be the matrix obtained from B by replacing column i ($1 \leq i \leq m$) with the $n \times f_i$ matrix corresponding to its X-adic expansion. If we choose e_i and f_i as in the proof of Theorem 40, we have $|\vec{e}|, |\vec{f}| < 2m$. At this point we have constructed $\bar{A}$ and $\bar{B}$ such that (71) holds.

Decompose $\bar{A}$ and $\bar{B}$ conformally into $t = \lceil n/m \rceil$ blocks so that

$$\bar{A}\bar{B} = \overset{\bar{A}}{\left[\ \bar{A}_1 \ \middle|\ \bar{A}_1 \ \middle|\ \cdots \ \middle|\ \bar{A}_t\ \right]} \overset{\bar{B}}{\begin{bmatrix} \bar{B}_1 \\ \hline \bar{B}_2 \\ \hline \vdots \\ \hline \bar{B}_t \end{bmatrix}}, \tag{72}$$

with each block $\bar{A}_*$ of column dimension m, except for possibly $\bar{A}_t$, which may have fewer than m columns. Then

$$\begin{aligned} \mathbf{c}\mathrm{mod}(AB, F) &= \mathbf{c}\mathrm{mod}\big(\sum_{i=1}^{t} \mathrm{Transpose}(\vec{X}^{(\vec{e})})\bar{A}_i\bar{B}_i\vec{X}^{(\vec{f})}, F\big) \\ &= \mathbf{c}\mathrm{mod}\big(\mathrm{Transpose}(\vec{X}^{(\vec{e})})\,\mathbf{c}\mathrm{mod}\big(\sum_{i=1}^{t} \bar{A}_i\bar{B}_i\vec{X}^{(\vec{f})}, F\big), F\big). \end{aligned}$$

The complete recipe to compute $C = \mathbf{c}\mathrm{mod}(AB, F)$ using partial linearization is as follows:

step 1 :	Construct $\bar{A}$ such that $A = \vec{X}^{(\vec{e})}\bar{A}$	: row linearization
step 2 :	Construct $\bar{B}$ such that $B = \bar{B}\vec{X}^{(\vec{f})}$	: column linearization
step 3 :	Decompose $\bar{A}$ and $\bar{B}$ as in (72)	: block decomposition
step 4 :	$\bar{C}_i := \bar{A}_i\bar{B}_i$, $1 \le i \le t$	: products over $\mathbb{Z}$
step 5 :	$\hat{C}_i := \mathbf{c}\mathrm{mod}\big(\bar{C}_i\vec{X}^{(\vec{f})}, F\big)$, $1 \le i \le t$	: column compressions
step 6 :	$\hat{C} := \mathbf{c}\mathrm{mod}\big(\sum_{i=1}^{t}\hat{C}_i, F\big)$	: modular matrix sum
step 7 :	$C := \mathbf{c}\mathrm{mod}(\mathrm{Transpose}(\vec{X}^{(\vec{e})})\hat{C}, F)$	: row compression

We now give a cost analysis in terms of bit operations. Similar to the proof of Theorem 40, the "products over $\mathbb{Z}$" step 4 has cost $O(nm^{\omega-1}\,\mathsf{M}(\log X + \log m))$. Next we show that the other steps are dominated by this cost.

Like in the the proof of Theorem 40, the "row linearization" step 1 and "column linearization" step 2 can both be done in linear time $O(nm\log X)$. The "block decomposition" step 3 does not involve any computation. Similar to the proof of Theorem 40, the "column compressions" step 5 can be done in time $O(nm^{\omega-1-\epsilon_0}\,\mathsf{M}(D/m + \log X + \log m)$, which dominates the cost of the "modular matrix sum" step 6.

Finally, consider the "row compression" step 7. Using a procedure with computational steps identical to those in the recipe for the "row expansion" step in the proof of Theorem 40, but incorporating here a Horner type scheme to accumulate the results using some additional summations, gives the same cost bound of $O(m^{\omega-\epsilon_0}\,\mathsf{M}(D/m + \log X))$ as derived in Theorem 40. □

8. Complexity analysis of Algorithm `HermiteBasis`

Algorithm `HermiteBasis`(S, F, k, m) takes as input a reduced Smith massager (S, F) that has an index (k, m) Hermite basis. By Corollary 22, such an S will have at most m nontrivial invariant factors, and so by Lemma 18 the additional precondition of the algorithm that S and F have column dimension m is not an essential restriction. In this section we establish the following result.

Theorem 44. *If, up to trivial invariant factors in S and corresponding leading zero columns in F, $[\epsilon](S, F, k, m)$ is a valid input to Algorithm `HermiteBasis`, then the Hermite basis of $\mathcal{R}(S, F)$ can be computed in $O(nm^{\omega-1}\,\mathsf{B}((\log\det S)/m + \log m)(\log m)^2\log(1/\epsilon))$ bit operations. The algorithm is randomized of the Las Vegas type: it either returns the correct output, or reports* FAIL *with probability at most ϵ.*

We defer the proof of Theorem 44 to the end of this section.

For convenience, we will initially assume that m is a power of two. The recursion tree for Algorithm `HermiteBasis` is then a perfect binary tree of height $\log_2 m$. Lemma 45 bounds the total cost of all work except for the calls to the `SmithMassager`. Lemma 46 then bounds the total cost of all calls to the randomized algorithm `SmithMassager`. Lemma 47 shows that assuming m to be a power of two is not an essential restriction, with the proof of Theorem 44 then following.

Lemma 45. *Not counting the cost of the calls to `SmithMassager`, Algorithm `HermiteBasis` has running time bounded by $O(nm^{\omega-1}\,\mathsf{B}((\log\det S)/m + \log m))$ bit operations. This result assumes that m is a power of two.*

Proof. We will prove that the sum of the cost of the nonrecursive work over all nodes in the recursion tree, not counting the cost of the calls to `SmithMassager`, satisfies the bound stated in the theorem. For brevity, let $D := \log\det S$. When discussing a particular node, we will distinguish the current value of a parameter by using a bar, for example, a particular node corresponds to the calling sequence $(\bar{S}, \bar{F}, *, \bar{m})$.

The internal nodes correspond to calling sequences with $\bar{m} > 1$. Each internal node has two children corresponding to the two recursive calls. The cost of the nonrecursive work at a particular internal node depends on n (fixed for all nodes), $\bar{m}$ (decreasing by a factor of 2 for the subproblems) and $\bar{D} := \log\det\bar{S}$ (splitting into $\bar{D} = \bar{D}_1 + \bar{D}_2$ corresponding to $\det\bar{S} = (\det\bar{S}_1)(\det\bar{S}_2)$ for the subproblems). Level i of the tree has 2^i nodes, each with $\bar{m} = m/2^i$. The sum of $\bar{D}$ over all nodes at any particular level is equal to D.

Consider the top level of the recursion in Algorithm `HermiteBasis`. We claim that, ignoring the two calls to `SmithMassager`, and up to a multiplicative constant, the cost of the nonrecursive work in the "`else`" case is bounded by

$$n\bar{m}^{\omega-1}\,\mathsf{B}(\bar{D}/\bar{m}+\log_2\bar{m}). \tag{73}$$

Theorem 33 gives the bound for computing T and (K, C). Theorem 40 gives the bound for computing $\mathbf{c}\bmod(FV_1, S_1)$ and $\mathbf{c}\bmod(CV_2, S_2)$. Corollary 42 gives the bound for computing $\mathbf{c}\bmod(H_1F, S)$. (In fact, for these three partially linearized multiplications the slightly better bound (73) with B replaced by M also holds.)

At this point we have established that each internal node of the recursion tree has associated cost bounded by (73). We now bound the sum of (73) over all nodes in a non-leaf level i. Any non-leaf level has $\bar{m} \geq 2$ and thus the argument of B in (73) satisfies $\bar{D}/\bar{m} + \log_2 m \geq 1$. The superlinearity of B gives that the sum of (73) over all nodes at a non-leaf level i is bounded by

$$\begin{aligned} n\bar{m}^{\omega-1}\,\mathsf{B}(D/\bar{m}+2^i\log_2\bar{m}) &= n(m/2^i)^{\omega-1}\,\mathsf{B}(D/(m/2^i)+2^i\log_2\bar{m}) \\ &= nm^{\omega-1}(1/2^i)^{\omega-1}\,\mathsf{B}(2^i\times(D/m+\log_2\bar{m})) \\ &\leq nm^{\omega-1}(1/2^i)^{\omega-1}\,\mathsf{B}(2^i)\mathsf{B}(D/m+\log_2\bar{m}) \\ &\in O(nm^{\omega-1}\mathsf{B}(D/m+\log\bar{m}))\times\frac{1}{(2^{\epsilon_0})^i}. \end{aligned} \tag{74}$$

Here, we used $\mathsf{B}(2^i) \in O((2^i)^{\omega-1-\epsilon_0})$ to obtain (74).

If we substitute the global upper bound $\log m$ for $\log\bar{m}$ in (74), then the only term in (74) that depends on i is $1/(2^{\epsilon_0})^i$. Summing $(1/2^{\epsilon_0})^i$ over all levels $i \geq 0$ yields a constant, so the total cost at all internal nodes is $O(nm^{\omega-1}\,\mathsf{B}(D/m+\log m))$, which is bounded by the cost stated in the theorem.

It remains to bound the cost of the base cases corresponding to $\bar{m} = 1$. The base cases coincide with calling sequences $(h_iI_1, *, k+i-1, 1)$ for $1 \leq i \leq m$, with the h_i being the diagonal entries of the block $\bar{H}$ in the index (k, m) Hermite basis H of $\mathcal{R}(S, F)$, see (31). Lemma 26 bounds the cost of a base case by $O(n\,\mathsf{B}((\log h_i)+1))$, where we have added $+1$ to ensure the argument of B is bounded from below by 1. The superlinearity of B then gives

$$\begin{aligned} \sum_{i=1}^{m} n\,\mathsf{B}((\log h_i)+1) &\leq n\,\mathsf{B}(D+m) \\ &= n\,\mathsf{B}(m((D/m)+1)) \\ &\leq n\,\mathsf{B}(m)\mathsf{B}(D/m+1) \\ &\in O(nm^{\omega-1-\epsilon_0}\,\mathsf{B}(D/m+1)). \quad \square \end{aligned}$$

Lemma 46. *The total cost of all calls to `SmithMassager` in Algorithm `HermiteBasis` is bounded by* $O(m^\omega\,\mathsf{B}((\log\det S)/m+\log m)(\log m)^2\log(1/\epsilon))$ *bit operations. This result assumes that m is a power of two.*

Proof. Consider the top level of the recursion. The two calls to `SmithMassager`$[\epsilon/4]$ are with arguments T and K. Both $\det T$ and $\det K$ are divisors of $\det S$. By Theorem 8, the calls to `SmithMassager`$[\epsilon/4]$ have cost bounded by that stated in the lemma. At level i of the recursion, the calls to `SmithMassager` should return FAIL with probability at most $\epsilon/4^{i+1}$, which increases the $\log(1/\epsilon)$ factor to $(i+\log(1/\epsilon))$.

Similar to the derivation of the bound (74) in the proof of Lemma 45, the total cost of the calls to `SmithMassager` at level i is bounded by

$$m^\omega\,\mathsf{B}((\log\det S)/m+\log\bar{m})(\log\bar{m})^2\times\frac{i+\log(1/\epsilon)}{(2^{\epsilon_0})^i}, \tag{75}$$

where $\bar{m} = m/2^i$. To arrive at (75) it suffices to replace n with m in (74) and to multiply by the missing factor $(\log\bar{m})^2(i+\log(1/\epsilon))$. The result now follows by substituting the global upper bound $\log m$ for $\log\bar{m}$

into (75) and then noting that

$$\sum_{i\geq 0} \frac{i+\log(1/\epsilon)}{(2^{\epsilon_0})^i} \in O(\log(1/\epsilon)). \quad \square$$

Lemma 47. *If (S, F, k, m) is a valid calling sequence to Algorithm* `HermiteBasis`*, then the calling sequence $(\mathrm{diag}(I_t, S), \mathrm{diag}(\mathbf{0}_t, F), k, m+t)$ is also valid, $t \in \mathbb{Z}_{\geq 0}$. Moreover, the Hermite basis of $\mathcal{R}(S, F)$ is the trailing submatrix of the Hermite basis of $\mathcal{R}(\mathrm{diag}(I_t, S), \mathrm{diag}(\mathbf{0}_t, F))$.*

Proof. Let H be the Hermite basis of $\mathcal{R}(S, F)$. The result follows from Lemma 13 by noting that

$$\left[\begin{array}{cc|cc} I_t & & & \\ & S & & \\ \hline \mathbf{0}_t & & I_t & \\ & F & & I_n \end{array}\right] \text{ has Hermite form } \left[\begin{array}{cc|cc} I_t & & & \\ & I_m & & * \\ \hline & & I_t & \\ & & & H \end{array}\right] \quad \square$$

Proof of Theorem 44. By working with the input $[\epsilon](\mathrm{diag}(I_t, S), \mathrm{diag}(\mathbf{0}_t, F), k, m+t)$ of Lemma 47, where t is at most $m-1$, we may assume without loss of generality that m is a power of 2. The result now follows as a corollary of Theorem 28 (correctness), Theorem 29 (probability), and Lemmas 45 and 46 (running time). $\square$

9. Hermite basis of an arbitrary relations lattice

In this section we show how to compute the Hermite basis H of $\mathcal{R}(M, G)$ with arbitrary inputs. Lemma 48 shows how to construct new inputs that satisfy the preconditions of Algorithm `HermiteBasis`. Theorem 49 then gives overall cost estimates for computing H.

Lemma 48. *Let $M \in \mathbb{Z}^{\ell\times m}$ have full column rank and $G \in \mathbb{Z}^{n\times m}$. There exists a randomized algorithm $(S, F) \leftarrow$* `ToSmithCoprime`$[\epsilon](M, G)$ *that computes matrices S and F such that $\mathcal{R}(S, F) = \mathcal{R}(M, G)$, with S a nonsingular Smith form, F reduced column-modulo S, and S and F coprime. The cost of the algorithm is $O((\ell+n)m^{\omega-1}\,\mathsf{B}(D/m+\log m)(\log m)^2\log(1/\epsilon))$ bit operations, where D is a bound for the sum of the bitlengths of the columns of M and G. The algorithm either returns a correct output, or reports* FAIL *with probability at most ϵ.*

Proof. We will perform a sequence of six transformations from the input (M, G) to (S, F):

$$(M, G) \xrightarrow{1} \left(\left[\frac{M_1}{M_2}\right], G\right) \xrightarrow{2} \left(\left[\frac{S_1}{M_3}\right], G_1\right) \xrightarrow{3} (T_1, G_1) \xrightarrow{4} (S_2, G_2) \xrightarrow{5} (K, C) \xrightarrow{6} (S, F).$$

Considering transformations 1 to 6 in turn we have: M_1 is a nonsingular submatrix of M, S_1 is the Smith form of M_1, T_1 is the Hermite basis of the previous modulus, S_2 is the Smith form of T_1, (K, C) are coprime inputs, and S is the Smith form of K. We will show how to do each transformation within the allotted time such that the transformed inputs generate the same relations lattice.

Before detailing the transformations, we explain how the cost estimate for each depends on a common upper bound $\bar{D} := D + (m/2)\log_2 m$ that captures the bitlength of the inputs. By assumption, D, and thus also $\bar{D}$, bounds the sum of the bitlengths of the columns of M and G. By Hadamard's bound, the base 2 logarithm of the absolute value of any nonzero minor of M is bounded by $\bar{D}$. In particular, $\log_2 |\det M_1| \leq \bar{D}$. The bitlength of subsequent inputs is then bounded by $\bar{D}$ by noting that $\det S_1 = |\det M_1|$, $(\det T_1) \mid (\det S_1)$, $\det S_2 = \det T_2$, and $(\det K) \mid (\det S_2)$. Moreover, M_3 and G_1 will be reduced column-modulo S_1, G_2 will be reduced column-modulo S_2, and C will be reduced column-modulo the diagonals of K.

Transformation 1 replaces M with PM, where P is a permutation matrix such that PM has principal $m \times m$ submatrix M_1 nonsingular. Correctness follows from Lemma 14.1. A suitable P can be found in the allotted time, with a chance of failure bounded by $\epsilon/4$, by randomly choosing primes $p \in \Theta(\bar{D})$ and computing a rank revealing matrix decomposition [17, 19, 11] of A modulo p. See, for example, [30, Proof of Theorem 1].

For the remaining transformations, the second matrix in each pair has row dimension n, and M_2 and M_3 have row dimension $\ell - m$; the cost estimate in the theorem uses the common upper bound $\ell + n$ for these row dimensions. All remaining matrices M_1, S_1, T_1, S_2 and K are square of dimension m.

Transformations 2 to 6 are accomplished as follows. For each transformation we cite the lemma that shows correctness. For each step, the results showing that the computation can be done in the allotted time is cited to the right. As noted above, a valid argument for the function M or B in the cost estimates given by the results cited to the right is $\bar{D}/m + \log m \in O(D/m + \log m)$.

$\xrightarrow{2}$: Lemma 17
$(S_1, V_1) := \texttt{SmithMassager}[\epsilon/4](M_1)$ *# Theorem 8*
$M_3 := \mathbf{c}\mathrm{mod}(M_2 V_1, S_1)$ *# Corollary 41*
$G_1 := \mathbf{c}\mathrm{mod}(G V_1, S_1)$ *# Corollary 41*

$\xrightarrow{3}$: Lemma 15.1
$T_1 :=$ the Hermite basis of $\mathcal{L}(S_1) + \mathcal{L}(M_3)$ *# Theorem 33.1*

$\xrightarrow{4}$: Lemma 16
$(S_2, V_2) := \texttt{SmithMassager}[\epsilon/4](T_1)$ *# Theorem 8*
$G_2 := \mathbf{c}\mathrm{mod}(G_1 V_2, S_2)$ *# Theorem 40*

$\xrightarrow{5}$: Lemma 20
$T_2 :=$ the Hermite basis of $\mathcal{L}(S_2) + \mathcal{L}(G_2)$ *# Theorem 33.1*

$$\begin{bmatrix} T_2 & & * \\ & I & C \\ & & K \end{bmatrix} := \text{the Hermite basis of } \begin{bmatrix} T_2 & & I \\ G_2 & I & \\ S_2 & & \end{bmatrix} \qquad \text{\# Theorem 33.2}$$

$\xrightarrow{6}$: Lemma 16
$(S, V_3) := \texttt{SmithMassager}[\epsilon/4](K)$ *# Theorem 8*
$F := \mathbf{c}\mathrm{mod}(C V_3, S)$ *# Theorem 40*

Including step 1, the above recipe has four Las Vegas steps, each with probability of failure bounded by $\epsilon/4$. The overall probability of failure is thus bounded by ϵ, as required. □

Theorem 49. *Let $M \in \mathbb{Z}^{\ell \times m}$ have full column rank and $G \in \mathbb{Z}^{n \times m}$. There exists a Las Vegas randomized algorithm that computes the Hermite basis H of a $\mathcal{R}(M, G)$ in*

$$O((\ell + n) m^{\omega - 1}\, \mathsf{B}(D/m + \log m)(\log m)^2) \tag{76}$$

plus

$$O(n^{\omega}\, \mathsf{B}(D/n + \log n)(\log n)^2) \tag{77}$$

bit operations, where D is a bound for the sum of the bitlengths of the column of M and G.

Moreover, if $n \in O(m)$, or the Hermite basis H of $\mathcal{R}(M, G)$ is $(k, \bar{m})$ for a known $(k, \bar{m})$ with $\bar{m} \in O(m)$, then the running time is (76), without requiring the extra term (77).

Proof. The algorithm for the general case has two steps:

$$(S, G) := \texttt{ToSmithCoprime}[1/4](M, G)$$
$$H := \texttt{HermiteBasis}[1/4](S, G, 0, n)$$

By Lemma 48, (S, G) can be computed in time (76). From the proof of Lemma 48 we have that $\log_2 \det S \leq D + (m/2) \log_2 m$. Using this bound for $\log \det S$, Theorem 44 states that H can be computed in time (77). Since both steps are Las Vegas with each having a probability of failure bounded by $1/4$, the overall probability of failure is bounded by $1/2$.

Now consider the case where H is index $(k, \bar{m})$ for a known $(k, \bar{m})$ with $\bar{m} \in O(m)$. Then we modify the second step to be

$$H := \texttt{HermiteBasis}[1/4](S, G, k, \bar{m}).$$

To capture the assumption that $\bar{m} \in O(m)$, let $\bar{m} = cm$ for some constant c. Then by Theorem 44 the call the `HermiteBasis` has cost $O(n(cm)^{\omega-1}\mathsf{B}(D/(cm) + \log cm)(\log cm)^2)$, which is bounded by (76) when c is a constant.

Finally, if $n \in O(m)$ then we know that H is index $(0, \bar{m})$ for $\bar{m} = n \in O(m)$, so this case has already been handled. □

10. Applications

In this section we give five examples of the use of our algorithm to compute the Hermite basis of a relations lattice. The problems we consider take as input one or more integer matrices. In terms of these input matrices we define M and F such that $\mathcal{R}(M, F)$ is the lattice we seek. We then use Theorem 49 to bound the cost of computing the Hermite basis of $\mathcal{R}(M, F)$.

10.1. Hermite basis of a full column rank matrix

The Hermite basis of a full column rank A is the Hermite basis of $\mathcal{R}(A, I)$. Using Theorem 49 to compute the Hermite basis of $\mathcal{R}(A, I)$ gives the following.

Theorem 50. *Let $A \in \mathbb{Z}^{n\times m}$ have full column rank. There exists a Las Vegas randomized algorithm that computes the Hermite form of A in $O(nm^{\omega-1}\,\mathsf{B}(D/m + \log m)(\log m)^2))$ bit operations, where D is a bound for the sum of the bitlengths of the columns of A.*

10.2. Remainder modulo a Hermite form

As done in [26] for polynomial matrices, we can compute the remainder of a matrix with respect to a Hermite basis, but now over $\mathbb{Z}$. Recall that the remainder of $F \in \mathbb{Z}^{n\times m}$ with respect to a Hermite basis $T \in \mathbb{Z}^{m\times m}$ is the matrix $\bar{F}$ of Lemma 15.2. Here, we observe that

$$\mathcal{R}\left(T, \left[\frac{-F}{I}\right]\right) \quad \text{has Hermite basis} \quad H = \left[\begin{array}{c|c} I & \bar{F} \\ \hline & T \end{array}\right].$$

We obtain the following result using Theorem 49 to compute H, using the fact that H is index (n, m).

Theorem 51. *Let $T \in \mathbb{Z}^{m\times m}$ be a Hermite basis and let $F \in \mathbb{Z}^{n\times m}$. There exists a Las Vegas randomized algorithm that computes the remainder of F with respect to T in*

$$O((n+m)m^{\omega-1}\,\mathsf{B}(D/m + \log m)(\log m)^2)$$

bit operations, where D is bound for the sum of the bitlengths of the columns of T and F.

10.3. Hermite form of a product of matrices

Suppose A and B are nonsingular. An obvious way to compute the Hermite form H of the product AB is to compute $C = AB$ explicitly, and then compute the Hermite form of C. But the total size of C can be large. For example, consider the $(2n+1) \times (2n+1)$ matrices

$$A = \begin{bmatrix} I_n & 2^n v & \\ & 2^n + 1 & \\ & & I_n \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} I_n & & \\ & 1 & w \\ & & 2I_n \end{bmatrix},$$

where $v \in \mathbb{Z}^{n\times 1}$ and $w \in \mathbb{Z}^{1\times n}$ are the column and row vector of all ones, respectively. Both A and B are in Hermite form, and the sum of the bitlengths of all entries in A and B, as well as the Hermite form H of AB, is $O(n^2)$. But $C = AB$ contains the $n \times n$ submatrix $2^n vw$ filled with 2^n, and so the sum of the bitlength

of all entries in C is $\Omega(n^3)$. Instead of computing C explicitly, we can use the fact that the Hermite basis of AB is equal to the Hermite basis of $\mathcal{R}(AB, I)$. Up to transformations allowed by Lemma 14 we have

$$\mathcal{R}(AB, I) = \mathcal{R}\left(\left[\begin{array}{c|c} A & \\ I & B \end{array}\right], \left[\begin{array}{c|c} \mathbf{0} & I \end{array}\right]\right). \tag{78}$$

Note that identity (78) applies also to rectangular matrices A and B. Using Theorem 49 to compute the Hermite basis of the right hand side of (78) gives the following.

Theorem 52. *Let $A \in \mathbb{Z}^{n\times m}$ and $B \in \mathbb{Z}^{m\times *}$ be such that AB has full column rank. There exists a Las Vegas randomized algorithm that computes the Hermite basis of AB in*

$$O((n+m)m^{\omega-1}\,\mathsf{B}(D/m + \log m)(\log m)^2)$$

bit operations, where D is a bound for the sum of the bitlengths of the columns of A and B.

10.4. Intersection of bases

Suppose $A, B \in \mathbb{Z}^{n\times m}$ have full column rank. Then

$$\mathcal{L}(A) \cap \mathcal{L}(B) = \mathcal{R}\left(\left[\begin{array}{c|c} A & \\ & B \end{array}\right], \left[\begin{array}{c|c} I & I \end{array}\right]\right). \tag{79}$$

Using Theorem 49 to compute the Hermite basis of the right hand side of (79) gives the following.

Theorem 53. *Let $A, B \in \mathbb{Z}^{n\times m}$ have full column rank. There exists a Las Vegas randomized algorithm that computes the Hermite basis of $\mathcal{L}(A) \cap \mathcal{L}(B)$ in $O(nm^{\omega-1}\,\mathsf{B}(D/m + \log m)(\log m)^2)$ bit operations, where D is a bound for the sum of the bitlengths of the columns of A and B.*

10.5. Multivariable Chinese remainder problem

The Chinese remainder problem asks to find an integer x satisfying

$$x\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} \bmod \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \tag{80}$$

$$\vec{b} \qquad\qquad M$$

for a given vector $\vec{b} \in \mathbb{Z}^{1\times n}$ of remainders and a diagonal matrix $M \in \mathbb{Z}_{\geq 0}^{n\times n}$ of nonzero moduli. In the classical setting one typically makes the assumption that the moduli d_i are pairwise relatively prime — in which case there is a unique solution for x in the range $[0, \det M)$ — but the case of arbitrary moduli has also been considered [27]. Another natural generalization of (80) takes as input an $A \in \mathbb{Z}^{n\times n}$ and asks for a vector $\vec{x} \in \mathbb{Z}^{1\times n}$ (if it exists) to the linear diophantine system $\vec{x}A = \vec{b} \bmod M$. Knill [22] and Sury [37] give additional background and history, referring to this linear diophantine system as the *multivariable Chinese remainder problem*.

Here we observe that

$$\mathcal{R}\left(M, \left[\begin{array}{c} -\vec{b} \\ \hline A \end{array}\right]\right) \text{ has Hermite basis } \left[\begin{array}{c|c} h & \vec{x}_p \\ \hline & \bar{H} \end{array}\right], \tag{81}$$

where $\vec{x}_p \in \mathbb{Z}^{1\times n}$ is a particular solution to the scaled system $\vec{x}A = h\vec{b} \bmod M$, and the scaling factor $h \in \mathbb{Z}_{>0}$ is minimal to achieve consistency. Using Theorem 49 to compute the Hermite basis of (81) gives the following.

Theorem 54. *Let $M \in \mathbb{Z}_{\geq 0}^{n\times n}$ be nonsingular and diagonal, $A \in \mathbb{Z}^{n\times n}$ and $\vec{b} \in \mathbb{Z}^{1\times n}$. There exists a Las Vegas randomized algorithm that computes:*

(i) The minimal $h \in \mathbb{Z}_{>0}$ such that $\vec{x}A = h\vec{b} \bmod M$ has a solution for $\vec{x} \in \mathbb{Z}^{1\times n}$.

(ii) A particular solution vector $\vec{x}_p \in \mathbb{Z}^{1\times n}$ such that $\vec{x}_p A = h\vec{b} \bmod M$.

(iii) A Hermite basis $\bar{H} \in \mathbb{Z}^{n\times n}$ such that $\{\vec{x} \mid \vec{x}A = h\vec{b} \bmod M\} = \{\vec{x}_p + v\bar{H} \mid v \in \mathbb{Z}^{1\times n}\}$.

If A and $\vec{b}$ are reduced column-modulo M, then the cost of the algorithm is

$$O(n^\omega \, \mathsf{B}((\log \det M)/n + \log n)(\log n)^2)$$

bit operations.

Remark 55. *While computing a basis for a lattice of integer relations has not had considerable attention to date, the same is not true for the case of polynomials. In this case the CRT corresponds to polynomial interpolation while modules of polynomial relations define such well known problems as rational interpolation, multi-point Padé approximation and more generally M-Padé approximation along with their matrix generalizations [3, 26].*

11. Conclusion and topics for future research

Let $M \in \mathbb{Z}^{n\times n}$ be a nonsingular integer matrix and $d = \log n + \log ||M||$. In this paper we have given a Las Vegas randomized algorithm to compute the Hermite form of M using

$$O(n^\omega (\log n)^2 \, d^{1+\epsilon}) \tag{82}$$

bit operations, where ω is the exponent of matrix multiplication and the $+\epsilon$ captures the cost of integer arithmetic, which can be pseudo-linear. We make the common assumptions that $\omega > 2$ and $\epsilon < \omega - 2$.

Our approach is to solve a more general problem, that of computing the basis in Hermite form of an integer relations lattice. To compute the Hermite form H of M, we start with a Smith massager (S, F) for M that can be computed in cost (82) using an existing Las Vegas randomized algorithm. We then give a divide and conquer algorithm to compute the Hermite basis of the relations lattice $\mathcal{R}(S, F)$, which is equal to H, also in Las Vegas cost (82). The divide and conquer algorithm itself is randomized because it uses Smith massager computations during the construction of the inputs for the recursive calls.

In terms of complexity, the main open problem concerns the need for randomization: find (deterministic) algorithms to compute the Hermite form H of M and the Smith form S of M using

$$(n^\omega d)^{1+o(1)} \tag{83}$$

bit operations. Regarding the Smith form, on the one hand, the existence of an algorithm for the more general problem of computing a Smith massager (S, F) of M in cost (83) would answer positively the question about computing H. On the other hand, finding just the largest invariant factor of S in cost (83) is open.

A natural analogue to the problems we consider is computing Hermite and Smith forms of matrices over the ring of univariate polynomials $\mathsf{K}[x]$, K a field. Over $\mathsf{K}[x]$ the cost is the number of required field operations from K and d is a bound for the degrees of entries in the input matrix. Given a nonsingular $M \in \mathsf{K}[x]^{n\times n}$, deterministic algorithms with cost (83) are known for computing the Hermite form [23] and for computing the largest invariant factor [42]. However, the problem of computing the Smith form in cost (83) remains open.

Our algorithm constructively reduces the problem of computing the Hermite form to that of multiplying integer matrices, and is thus well suited to take advantage of hardware and software support both for matrix multiplication and fast integer arithmetic. At present we have a preliminary C implementation for the computation of a Smith massager [38], one of the core subroutines needed. It remains to implement the fast Hermite and Howell constructions along with the specialized column-modulo matrix multiplication operations. Such implementations require developing techniques to bridge the gap between an algorithm with good asymptotic complexity and one having practical performance.

References

[1] J. Alman and V. V. Williams. A refined laser method and faster matrix multiplication. In *Proc. of 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539, 2021.

[2] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix–type Padé approximants. *SIAM Journal on Matrix Analysis and Applications*, 15(3):804–823, 1994.

[3] B. Beckermann and G. Labahn. Recursiveness in matrix rational interpolation problems. *Journal of Computational and Applied Math*, 77:5–34, 1997.

[4] J.-F. Biasse, C. Fieker, and T. Hofmann. On the computation of the HNF of a module over the ring of integers of a number field. *Journal of Symbolic Computation*, 40(3):581–615, 2017.

[5] 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.

[6] S. Birmpilis, G. Labahn, and A. Storjohann. A fast algorithm for computing the Smith normal form with multipliers for a nonsingular integer matrix. *Journal of Symbolic Computation*, 116:146–182, 2023.

[7] S. Birmpilis, G. Labahn, and A. Storjohann. A cubic algorithm for computing the Hermite normal form of a nonsingular integer matrix. *ACM Transactions of Algorithms*, 19:1–36, 2023.

[8] Stavros Birmpilis, George Labahn, and Arne 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. ISBN 9781450360845.

[9] T-W. J. Chou and G. E. Collins. Algorithms for the solutions of systems of linear diophantine equations. *SIAM Journal of Computing*, 11:687–708, 1982.

[10] P. D. Domich, R. Kannan, and L. E. Trotter, Jr. Hermite normal form computation using modulo determinant arithmetic. *Mathematics of Operations Research*, 12(1):50–59, 1987.

[11] J.-G. Dumas, C. Pernet, and Z. Sultan. Fast computation of the rank profile matrix and the generalized Bruhat decomposition. *Journal of Symbolic Computation*, 83:187–210, 2017.

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

[13] 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.

[14] J. L. Hafner and K. S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM Journal of Computing*, 20(6):1068–1083, December 1991.

[15] C. Hermite. Sur l'introduction des variables continues dans la théorie des nombres. *J. Reine Angew. Math.*, 41:191–216, 1851.

[16] J. A. Howell. Spans in the module $(\mathbb{Z}_m)^s$. *Linear and Multilinear Algebra*, 19:67—77, 1986.

[17] 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.

[18] C. S. Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. *SIAM Journal of Computing*, 18(4):658–669, 1989.

[19] C.-P. Jeannerod, C. Pernet, and A. Storjohann. Rank-profile revealing Gaussian elimination and the CUP matrix decomposition. *Journal of Symbolic Computation*, 56:56–58, 2013.

[20] C.-P. Jeannerod, V. Neiger, É. Schost, and G. Villard. Computing minimal interpolation bases. *Journal of Symbolic Computation*, 83:272–314, 2017.

[21] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of and integer matrix. *SIAM Journal of Computing*, 8(4):499–507, November 1979.

[22] O. Knill. A Multivariable Chinese Remainder Theorem, 2012. URL `https://arxiv.org/abs/1206.5114`.

[23] G. Labahn, V. Neiger, and W. Zhou. Fast, deterministic computation of the Hermite normal form and determinant of a polynomial matrix. *Journal of Complexity*, 42:44–71, 2017.

[24] R. Liu and Y. Pan. Computing Hermite normal form faster via solving system of linear equations. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'19*, page 283–290, New York, NY, USA, 2019. ACM.

[25] D. Micciancio and B. Warinschi. A linear space algorithm for computing the Hermite normal form. In B. Mourrain, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'01*, pages 231—236. ACM Press, New York, 2001.

[26] V. Neiger and T. X. Vu. Computing canonical bases of modules of univariate relations. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'17*. ACM Press, New York, 2017.

[27] O. Ore. The General Chinese Remainder Theorem. *The American Mathematical Monthly.*, 59:365–370, 1952.

[28] C. Pauderis and A. Storjohann. Computing the invariant structure of integer matrices: Fast algorithms into practice. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'13*, pages 307–314, New York, NY, USA, 2013. ACM.

[29] C. Pernet and W. Stein. Fast computation of Hermite normal forms of integer matrices. *Journal of Number Theory*, 130(7):1675–1683, 2010.

[30] D. Saunders and Zhendong Wan. Smith normal form of dense integer matrices, fast algorithms into practice. In J. Gutierrez, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'04*, pages 274–281. ACM Press, New York, 2004.

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

[32] A. Storjohann. High–order lifting. Extended Abstract. In T. Mora, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'02*, pages 246–254. ACM Press, New York, 2002.

[33] A. Storjohann. High–order lifting and integrality certification. *Journal of Symbolic Computation*, 36 (3–4):613–648, 2003. Extended abstract in [32].

[34] A. Storjohann. Notes on computing minimal approximant bases. In W. Decker, M. Dewar, E. Kaltofen, and S. Watt, editors, *Challenges in Symbolic Computation Software*, number 06271 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. URL `http://drops.dagstuhl.de/opus/volltexte/2006/776`.

[35] A. Storjohann and G. Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. In Y. N. Lakshman, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'96*, pages 259–266. ACM Press, New York, 1996.

[36] A. Storjohann and T. Mulders. Fast algorithms for linear algebra modulo N. In G. Bilardi, G. F. Italiano, A. Pietracaprina, and G. Pucci, editors, *Algorithms — ESA '98*, LNCS 1461, pages 139–150. Springer Verlag, 1998.

[37] B Sury. Multivariable Chinese Remainder Theorem. *Resonance*, 20:206–216, 2015.

[38] Z. Wang, S. Birmpilis, G. Labahn, and A. Storjohann. A C implementation of the Smith massager algorithm, 2026. In progress.

[39] W. Zhou and G. Labahn. Efficient computation of order basis. In J. P. May, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'09*, pages 375–384. ACM Press, New York, 2009.

[40] W. Zhou and G. Labahn. Computing column bases of polynomial matrices. In *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'13*, pages 379–388. ACM Press, Boston, USA, 2013.

[41] W. Zhou, G. Labahn, and A. Storjohann. Computing minimal nullspace basis. In J. van der Hoeven and M. van Hoeij, editors, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC'12*, pages 366–373. ACM Press, New York, 2012.

[42] W. Zhou, G. Labahn, and A. Storjohann. A deterministic algorithm for inverting a polynomial matrix. *Journal of Complexity*, 31:162–173, 2015.