Computing a Basis for an Integer Lattice

by

Haomin Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

Abstract

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

Acknowledgements

My first acknowledgment must go to my supervisor **Prof. Arne Storjohann** for his invaluable advice and endless care. His flawless guidance and mental support “safely” led me through this two-year’s academia life during the pandemic. Thanks to him, I learned how to define problems, research approaches, and practice meticulous academic writing. His infinite enthusiasm for symbolic computation research and extensive knowledge of English literature inspired me all the time. I am really lucky to be able to study with Prof. Storjohann.

I also want to thank **Prof. George Labahn**. Three years ago, I took the Numerical Computation course with Prof. Labahn, which introduced me to the world of symbolic computation and was one of my most favorite courses during my undergraduate studies. I would also like to express my gratitude to the Instructional Support Coordinator **Karen Anderson** for offering me invaluable teaching opportunities at school and plenty of life advice. She taught me that “Don’t sell yourself short. Hard work will pay off in the near future”.

I gratefully appreciate my thesis committee members, **Prof. George Labahn** and **Prof. Eugene Zima**, for their time reading my thesis and their constructive feedback.

I want to greatly thank my parents for their unconditional support and absolute trust. In the critical moments of my life, in the highest highs and in the depths of true lows, I have never had to turn to them, because they were already there.

I want to give plenty of appreciation to my friends for their companionship during my graduate studies. Many thanks go to Symbolic Computation Group.

Table of Contents

List of Figures

Chapter 1

Introduction

A lattice is an infinite set of points in n-th real coordinate dimension space with a regular arrangement.

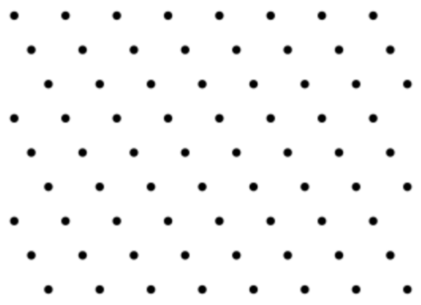

Figure 1.1: Lattice on 2D plane

It has the properties that two points in the lattice can produce another lattice point by coordinate-wise addition and subtraction. An integer lattice is a lattice whose lattice points are in integer space. Lattices have been studied since the late 18th century by mathematicians, including Lagrange, Gauss, and Minkowski. Lattices have applications in both computer science and mathematics, such as error encoding/decoding, the solution of integer programming problems, cryptanalysis, and more.

In this thesis, we represent integer lattices in algebraic form. Let an integer matrix

$\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ be comprised of n row vectors $\boldsymbol{a}_i \in \mathbb{Z}^{1\times m}$ where $1 \le i \le n$, that is,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_n \end{bmatrix} \in \mathbb{Z}^{n\times m}.$$

The integer lattice generated by the rows of $\boldsymbol{A}$ is the set

$$\mathcal{L}(\boldsymbol{A}) = \left\{ \sum_{i=1}^{n} c_i \boldsymbol{a}_i \mid c_i \in \mathbb{Z} \right\} = \left\{ \boldsymbol{c}\,\boldsymbol{A} \mid \boldsymbol{c} \in \mathbb{Z}^{1\times n} \right\}.$$

We consider the problem of computing a basis for $\mathcal{L}(\boldsymbol{A})$. If the rank of a matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ is r, then a basis for $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ is given by a matrix $\boldsymbol{B} \in \mathbb{Z}^{r\times m}$ such $\mathcal{L}(\boldsymbol{A}) = \mathcal{L}(\boldsymbol{B})$. Before continuing, we mention that $\mathcal{L}(\boldsymbol{A})$ will always denote the lattice generated by the *rows* of $\boldsymbol{A}$, and, for brevity, we will write "$\boldsymbol{B}$ is a basis for $\boldsymbol{A}$" to mean "$\boldsymbol{B}$ is a basis for $\mathcal{L}(\boldsymbol{A})$."

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

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

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

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

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

Another case is n arbitrary and $m = 1$. Given $\boldsymbol{A} \in \mathbb{Z}^{n\times 1}$, find $g = \gcd(\boldsymbol{A})$, the gcd of all entries in $\boldsymbol{A}$. Storjohann [2000, Corollary 6.5] gives an $O(n\,\mathsf{B}(\log ||\boldsymbol{A}||))$ bit operations algorithm that solves the vector extended gcd problem, producing a $\boldsymbol{C} \in \mathbb{Z}^{1\times n}$ that satisfies $||\boldsymbol{C}|| \le ||\boldsymbol{A}||$.

Example 1. *Let*

$$\boldsymbol{A} := \begin{bmatrix} 336 \\ -420 \\ 357 \\ 140 \\ -322 \end{bmatrix} \in \mathbb{Z}^{5\times 1},$$

where $\gcd(\boldsymbol{A}) = 7$. *The algorithm of Storjohann [2000, Corollary 6.5] produces*

$$\boldsymbol{C} = \begin{bmatrix} 21 & 0 & 1 & 0 & 23 \end{bmatrix} \in \mathbb{Z}^{1\times 5}$$

such that $\boldsymbol{C}\boldsymbol{A} = \gcd(\boldsymbol{A})$.

The general version of the problem takes as input an $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ with full column rank, and asks as output a $\boldsymbol{B} \in \mathbb{Z}^{m\times m}$ such that $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{A})$. Such a $\boldsymbol{B}$ is a basis for $\mathcal{L}(\boldsymbol{A})$. Chen and Storjohann [2005] give a randomized algorithm that can produce a $\boldsymbol{C}_1 \in \mathbb{Z}^{(m+1)\times n}$ to compress $\boldsymbol{A}$ such that $\mathcal{L}(\boldsymbol{C}_1\boldsymbol{A}) = \mathcal{L}(\boldsymbol{A})$. They show that if $\boldsymbol{C}_1 \in \mathbb{Z}^{(m+1)\times n}$ has entries chosen uniformly and randomly from $\{0, 1, \ldots, \lambda - 1\}$ where $\lambda \geq 1200m\log(2nm||\boldsymbol{A}||)$ and λ is a multiple of 30, then $\mathcal{L}(\boldsymbol{C}_1\boldsymbol{A}) = \mathcal{L}(\boldsymbol{A})$ with probability greater than 0.4.

Example 2. *Consider the full column rank matrix*

$$\boldsymbol{A} := \begin{bmatrix} -309 & 1215 & -747 & -5199 \\ 2471 & -3360 & 1318 & 3947 \\ -2 & -2120 & -486 & -236 \\ 3201 & -2402 & -890 & 3427 \\ 535 & -201 & 2306 & -59 \\ 1574 & -458 & 845 & 4620 \end{bmatrix} \in \mathbb{Z}^{6\times 4}.$$

We pick $\lambda = 86070$ *and randomly generate a matrix* $\boldsymbol{C}_1$ *from* $\{0, 1, \ldots, \lambda - 1\}$*:*

$$\boldsymbol{C}_1 = \begin{bmatrix} 36468 & 66377 & 81948 & 60787 & 63158 & 71845 \\ 72730 & 17619 & 25671 & 54005 & 61155 & 41887 \\ 42709 & 84439 & 51124 & 56075 & 59063 & 73771 \\ 311 & 50654 & 30932 & 32446 & 13213 & 10645 \\ 15545 & 22630 & 15747 & 1345 & 23270 & 50828 \end{bmatrix} \in \mathbb{Z}^{5\times 6}.$$

We have

$$\bar{\boldsymbol{A}} := \boldsymbol{C}_1\boldsymbol{A} = \begin{bmatrix} 494037806 & -544058002 & 172667505 & 589567786 \\ 292529705 & -186451821 & 84769921 & 60345497 \\ 522559774 & -520557416 & 203169738 & 628677706 \\ 252691902 & -320861930 & 62083966 & 350607272 \\ 147841898 & -121720449 & 105974413 & 242847608 \end{bmatrix} \in \mathbb{Z}^{5\times 4}.$$

For this example the compression is successful and $\mathcal{L}(\bar{\boldsymbol{A}}) = \mathcal{L}(\boldsymbol{A})$.

However, the technique of Chen and Storjohann [2005] cannot handle the final step, that is, compressing an $(m+1)\times m$ integer matrix with full column rank to an $m\times m$ integer matrix.

A classic method to do this final step is to compute a basis $\bar{\boldsymbol{H}} \in \mathbb{Z}^{m\times m}$ for a full column rank integer matrix $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(m+1)\times m}$. The Hermite normal form is a canonical form over the principal ideal domain $\mathbb{Z}$, and can be considered to be a generalization of the reduced row echelon form of a matrix over a field. In the case of full column rank matrices, a matrix $\boldsymbol{H}$ is in Hermite normal form if

1. $\boldsymbol{H}$ is upper triangular,
2. diagonal entries of $\boldsymbol{H}$ are positive integers, and
3. off-diagonal entries in each column of $\boldsymbol{H}$ are non-negative and strictly smaller than the diagonal entry.

The Hermite normal form of a matrix $\bar{\boldsymbol{A}}$ has the same dimensions as $\bar{\boldsymbol{A}}$. By Hermite basis of $\bar{\boldsymbol{A}}$, we mean the submatrix of the Hermite form restricted to its nonzero rows.

Example 3. *Consider the full column rank matrix:*

$$\bar{\boldsymbol{A}} := \begin{bmatrix} -309 & 1215 & -747 & -5199 \\ 2471 & -3360 & 1318 & 3947 \\ -2 & -2120 & -486 & -236 \\ 3201 & -2402 & -890 & 3427 \\ 535 & -201 & 2306 & -59 \end{bmatrix} \in \mathbb{Z}^{5\times 4}.$$

The Hermite normal form of $\bar{\boldsymbol{A}}$ is

$$\boldsymbol{H} = \begin{bmatrix} 1 & 3 & 5 & 57332297439 \\ & 7 & 13 & 23323233552 \\ & & 15 & 10443440106 \\ & & & 164009996654 \\ & & & 0 \end{bmatrix} \in \mathbb{Z}^{5\times 4}.$$

Therefore, the Hermite basis of $\bar{\boldsymbol{A}}$ is

$$\bar{\boldsymbol{H}} = \begin{bmatrix} 1 & 3 & 5 & 57332297439 \\ & 7 & 13 & 23323233552 \\ & & 15 & 10443440106 \\ & & & 164009996654 \end{bmatrix} \in \mathbb{Z}^{4\times 4}.$$

The Hermite basis $\bar{\boldsymbol{H}} \in \mathbb{Z}^{4\times 4}$ gives a basis for the matrix $\boldsymbol{A} \in \mathbb{Z}^{6\times 4}$ in previous Example 2.

By using the Hermite normal form, we can also obtain a basis generator $\boldsymbol{C}_2 \in \mathbb{Z}^{m\times(m+1)}$ such that $\boldsymbol{C}_2\bar{\boldsymbol{A}}$ is a basis for $\bar{\boldsymbol{A}}$. Given an integer matrix $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(m+1)\times m}$ with full column rank, we have

$$\boldsymbol{U}\,\bar{\boldsymbol{A}} = \begin{bmatrix} \bar{\boldsymbol{H}} \\ \hline \end{bmatrix}$$

where $\bar{\boldsymbol{H}} \in \mathbb{Z}^{m\times m}$ is the Hermite basis for $\bar{\boldsymbol{A}}$ and $\boldsymbol{U} \in \mathbb{Z}^{(m+1)\times(m+1)}$ is unimodular. Then, the top m rows of $\boldsymbol{U}$ is our solution $\boldsymbol{C}_2 \in \mathbb{Z}^{m\times(m+1)}$.

Example 4. *Consider the same input matrix $\bar{\boldsymbol{A}} \in \mathbb{Z}^{5\times 4}$ as in Example 3:*

$$\bar{\boldsymbol{A}} := \begin{bmatrix} -309 & 1215 & -747 & -5199 \\ 2471 & -3360 & 1318 & 3947 \\ -2 & -2120 & -486 & -236 \\ 3201 & -2402 & -890 & 3427 \\ 535 & -201 & 2306 & -59 \end{bmatrix} \in \mathbb{Z}^{5\times 4}.$$

The Hermite normal form algorithm produces

$$\boldsymbol{U} = \begin{bmatrix} -15906805 & -12537803 & 2023147 & 7267703 & 5244556 \\ -6471015 & -5100479 & 823033 & 2956560 & 2133528 \\ -2897525 & -2283840 & 368529 & 1323858 & 955329 \\ -45504458 & -35866786 & 5787601 & 20790656 & 15003058 \\ -2 & -5 & 3 & 3 & 4 \end{bmatrix} \in \mathbb{Z}^{5\times 5}$$

such that

$$\boldsymbol{H} := \boldsymbol{U}\bar{\boldsymbol{A}} = \begin{bmatrix} 1 & 3 & 5 & 57332297439 \\ 0 & 7 & 13 & 23323233552 \\ 0 & 0 & 15 & 10443440106 \\ 0 & 0 & 0 & 164009996654 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{Z}^{5\times 4}$$

is in Hermite normal form. Then, we can take $\boldsymbol{C}_2$ to be the top 4 rows of $\boldsymbol{U}$:

$$\boldsymbol{C}_2 := \begin{bmatrix} -15906805 & -12537803 & 2023147 & 7267703 & 5244556 \\ -6471015 & -5100479 & 823033 & 2956560 & 2133528 \\ -2897525 & -2283840 & 368529 & 1323858 & 955329 \\ -45504458 & -35866786 & 5787601 & 20790656 & 15003058 \end{bmatrix} \in \mathbb{Z}^{4\times 5}$$

such that $\mathcal{L}(\bar{\boldsymbol{A}}) = \mathcal{L}(\boldsymbol{C}_2\bar{\boldsymbol{A}})$. Note that $\bar{\boldsymbol{H}} := \boldsymbol{C}_2\bar{\boldsymbol{A}} \in \mathbb{Z}^{4\times 4}$ is the Hermite basis of $\bar{\boldsymbol{A}}$ in previous Example 3.

Although the Hermite normal form of $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(m+1)\times m}$ gives a basis for the lattice of $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, a problem with this method is that the best *a priori* bound for the magnitude of entries in the Hermite basis $\bar{\boldsymbol{H}} \in \mathbb{Z}^{m\times m}$ and the solution $\boldsymbol{U} \in \mathbb{Z}^{(m+1)\times(m+1)}$ is $m^{m/2}\,||\bar{\boldsymbol{A}}||^m$ (Hadamard's bound), where $||\cdot||$ for a matrix or vector denotes the largest entry in absolute value. More clearly, the bitlength of entries in $\bar{\boldsymbol{H}}$ and $\boldsymbol{U}$ can be over m times the bitlength of entries in $\bar{\boldsymbol{A}}$. Algorithms working with integer matrices typically have bit complexity (running time) bounds given in terms of not only the dimension of the matrix but also the bitlength of entries. This growth in the bitlength can adversely affect the complexity of algorithms that work with $\boldsymbol{U}$ and $\bar{\boldsymbol{H}}$.

In this thesis, our goal is to produce a basis $\boldsymbol{B} \in \mathbb{Z}^{m\times m}$ for $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(m+1)\times m}$ that has entries with bitlength not much larger than those of $\bar{\boldsymbol{A}}$. Formally, we require that $\log||\boldsymbol{B}|| \in O(\log n + \log||\bar{\boldsymbol{A}}||)$.

In this thesis, we present a method to do the final compressing step from full column rank $(m+1)\times m$ to a nonsingular $m\times m$ basis with compressing deterministically. Going forward, instead of assuming the input matrix is a full column rank $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(m+1)\times m}$, we will assume the input matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ has full column rank; this change of notation will be convenient for deriving magnitude, sparsity and complexity bounds.

To the best of our knowledge, the fastest previous algorithm to compute a basis $\boldsymbol{B} \in \mathbb{Z}^{(n-1)\times(n-1)}$ for a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ with $\log||\boldsymbol{B}|| \in O(\log n + \log||\boldsymbol{A}||)$ is that of Li and Nguyen [2019, Theorem 3.8], which solves the problem in the general case, that is, for n, m and r arbitrary. Applied to the special case $m = n - 1 = r$ that we consider here, their algorithm has cost bounded by $O(n^\omega \mathsf{B}(nd))$ bit operations, where $d = \log n + \log ||\boldsymbol{A}||$ and ω is the exponent of matrix multiplication. In this thesis, we give an algorithm that solves the problem in $O(n^\omega \mathsf{B}(d)(\log n)^2)$ bit operations, so about a factor of n faster than the previously known fastest algorithm. We remark that this cost estimate matches that of the Smith form algorithm of Birmpilis et al. [2020] and thus our technique can be used for the application given in the later paragraph, at least for the special case $m = n - 1 = r$.

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

Example 5. *Consider the same input matrix $\bar{\boldsymbol{A}} \in \mathbb{Z}^{5\times 4}$ as in Example 4 and rename the matrix $\bar{\boldsymbol{A}}$ to $\boldsymbol{A}$:*

$$\boldsymbol{A} := \begin{bmatrix} -309 & 1215 & -747 & -5199 \\ 2471 & -3360 & 1318 & 3947 \\ -2 & -2120 & -486 & -236 \\ 3201 & -2402 & -890 & 3427 \\ 535 & -201 & 2306 & -59 \end{bmatrix} \in \mathbb{Z}^{5\times 4}.$$

Our algorithm produces

$$\boldsymbol{C} = \begin{bmatrix} 1 & 3 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \in \mathbb{Z}^{4\times 5}$$

and a basis

$$\boldsymbol{B} := \boldsymbol{C}\boldsymbol{A} = \begin{bmatrix} 7104 & -8865 & 3207 & 6642 \\ -2 & -2120 & -486 & -236 \\ 3201 & -2402 & -890 & 3427 \\ 535 & -201 & 2306 & -59 \end{bmatrix} \in \mathbb{Z}^{4\times 4}$$

such that $\mathcal{L}(\boldsymbol{A}) = \mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{C}\boldsymbol{A})$. Note that the magnitudes of the entries in $\boldsymbol{C} \in \mathbb{Z}^{4\times 5}$ and the basis $\boldsymbol{B} \in \mathbb{Z}^{4\times 4}$ are much smaller than the magnitudes of the entries in $\boldsymbol{C}_2 \in \mathbb{Z}^{4\times 5}$ and the Hermite basis $\bar{\boldsymbol{H}} \in \mathbb{Z}^{4\times 4}$ shown in Example 4 respectively.

The sparsity of $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ can be useful in some situations. For example, it can be exploited to perform fast matrix multiplication. Suppose, in addition to the full column rank input matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, we have linearly dependent columns $\bar{\boldsymbol{A}} \in \mathbb{Z}^{n\times m}$. If $\boldsymbol{C} \in$

$\mathbb{Z}^{(n-1)\times n}$ is such that $\boldsymbol{C}\boldsymbol{A}$ is a basis for $\boldsymbol{A}$, then a basis for the rank $n-1$ augmented matrix $\left[\begin{array}{c|c} \boldsymbol{A} & \bar{\boldsymbol{A}} \end{array}\right] \in \mathbb{Z}^{n\times((n-1)+m)}$ is given by $\boldsymbol{C}\left[\begin{array}{c|c} \boldsymbol{A} & \bar{\boldsymbol{A}} \end{array}\right]$. The additional cost to compute $\boldsymbol{C}\bar{\boldsymbol{A}}$ is only $O(nm\,\mathsf{M}(d)\log n)$ bit operations, where M is a cost function for integer multiplication, and $d = \log n + \log ||\bar{\boldsymbol{A}}||$.

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

$$\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right] \in \mathbb{Z}^{n\times n} \tag{1.1}$$

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

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

Example 6. *Considering the same input matrix $\boldsymbol{A} \in \mathbb{Z}^{5\times 4}$ as in Example 5:*

$$\boldsymbol{A} := \begin{bmatrix} -309 & 1215 & -747 & -5199 \\ 2471 & -3360 & 1318 & 3947 \\ -2 & -2120 & -486 & -236 \\ 3201 & -2402 & -890 & 3427 \\ 535 & -201 & 2306 & -59 \end{bmatrix} \in \mathbb{Z}^{5\times 4}.$$

The step (1) computes a left kernel basis for $\boldsymbol{A}$:

$$\boldsymbol{w} := \begin{bmatrix} -2 & -5 & 3 & 3 & 4 \end{bmatrix} \in \mathbb{Z}^{1\times 5}.$$

The step (2) then gives an augmentation vector for $\boldsymbol{A} \in \mathbb{Z}^{5\times 4}$:

$$\boldsymbol{v} := \begin{bmatrix} -3 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{Z}^{5\times 1}.$$

such that

$$\begin{aligned}
&\boldsymbol{w}\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right] \\
&= \begin{bmatrix} -2 & -5 & 3 & 3 & 4 \end{bmatrix} \left[\begin{array}{cccc|c} -309 & 1215 & -747 & -5199 & -3 \\ 2471 & -3360 & 1318 & 3947 & 1 \\ -2 & -2120 & -486 & -236 & 0 \\ 3201 & -2402 & -890 & 3427 & 0 \\ 535 & -201 & 2306 & -59 & 0 \end{array}\right] \\
&= \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & 1 \end{array}\right]
\end{aligned}$$

The step (3) finally calculates a left kernel basis of $\boldsymbol{v} \in \mathbb{Z}^{5\times 1}$:

$$\boldsymbol{C} := \begin{bmatrix} 1 & 3 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \in \mathbb{Z}^{4\times 5},$$

which is shown in Example 5.

One of our motivations for studying the problem of computing a basis for a matrix $\boldsymbol{A}$ is to extend fast algorithms for computing normal forms of integer matrices to input matrices with arbitrary shapes and rank profiles. For example, the fastest algorithm for computing the Smith form of an integer matrix by Birmpilis et al. [2020] requires as input a square nonsingular input matrix.

The Smith normal form of a matrix is a canonical form over the principal ideal domain $\mathbb{Z}$ and can be obtained from the original matrix by multiplying on the left and right by unimodular square matrices. In the case of full column rank matrices, a matrix $\boldsymbol{S}$ is in Smith normal form if

1. $\boldsymbol{S}$ is diagonal,
2. diagonal entries of $\boldsymbol{S}$ are positive integers,
3. each diagonal entry of $\boldsymbol{S}$ except for the last divides the next.

The Smith normal form of a matrix $\boldsymbol{S}$ has the same dimensions as $\boldsymbol{A}$. There exists unimodular matrices $\boldsymbol{U}$, $\boldsymbol{V}$ such that $\boldsymbol{U}\boldsymbol{A}\boldsymbol{V} = \boldsymbol{S}$.

Example 7. *Consider an integer matrix*

$$\boldsymbol{A} := \begin{bmatrix} -18 & -27 & -113 & 57 \\ 35 & 28 & 63 & -56 \\ -25 & -27 & -92 & 57 \\ -28 & -28 & -84 & 56 \\ -24 & -29 & -97 & 55 \\ -31 & -29 & -76 & 55 \end{bmatrix} \in \mathbb{Z}^{6\times 4}.$$

The Smith normal form of $\boldsymbol{A}$ *is*

$$\boldsymbol{S} = \begin{bmatrix} 1 & & & \\ & 7 & & \\ & & 28 & \\ & & & \\ & & & \\ & & & \end{bmatrix} \in \mathbb{Z}^{6\times 4}.$$

Suppose $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ has rank r. By first producing a basis $\boldsymbol{B} \in \mathbb{Z}^{r\times m}$ for $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, then producing a basis $\bar{\boldsymbol{B}} \in \mathbb{Z}^{r\times r}$ for Transpose($\boldsymbol{B}$) $\in \mathbb{Z}^{m\times r}$, we can apply the Smith form algorithm with the nonsingular input $\bar{\boldsymbol{B}}$ to obtain the Smith form of $\boldsymbol{A}$.

Example 8. *Consider the integer matrix with rank* 3*:*

$$\boldsymbol{A} := \begin{bmatrix} -18 & -27 & -113 & 57 \\ 35 & 28 & 63 & -56 \\ -25 & -27 & -92 & 57 \\ -28 & -28 & -84 & 56 \\ -24 & -29 & -97 & 55 \\ -31 & -29 & -76 & 55 \end{bmatrix} \in \mathbb{Z}^{6\times 4}.$$

Firstly, we compute a basis for $\boldsymbol{A}$*:*

$$\boldsymbol{B} = \begin{bmatrix} 2 & 3 & -3 & 3 \\ 1 & -2 & -5 & -2 \\ -53 & -34 & -15 & 50 \end{bmatrix} \in \mathbb{Z}^{3\times 4}.$$

Then, we produce a basis for

$$\text{Transpose}(\boldsymbol{B}) = \begin{bmatrix} 2 & 1 & -53 \\ 3 & -2 & -34 \\ -3 & -5 & -15 \\ 3 & -2 & 50 \end{bmatrix} \in \mathbb{Z}^{4\times 3},$$

that is,

$$\bar{\boldsymbol{B}} = \begin{bmatrix} 26 & -1 & -171 \\ -3 & -5 & -15 \\ 3 & -2 & -34 \end{bmatrix} \in \mathbb{Z}^{3\times 3}.$$

The Smith form of $\bar{\boldsymbol{B}}$ is

$$\bar{\boldsymbol{S}} = \begin{bmatrix} 1 & & \\ & 7 & \\ & & 28 \end{bmatrix} \in \mathbb{Z}^{3\times 3.}$$

The Smith normal form $\bar{\boldsymbol{S}}$ is equal to the submatrix of the Smith normal form $\boldsymbol{S}$ of $\boldsymbol{A}$ restricted to its nonzero rows and columns in Example 7.

Computing a lattice with entries not much larger than those of the input matrix has many other applications, such as rational reconstruction, attacking GGH Public Key Cryptosystem, solving Shortest Vector Problem, and so on.

The rest of this thesis is organised as follows.

In Chapter 2, we will provide the fundamental mathematical concepts for lattice, primitive matrix, and left kernel matrix. We will also recall known theorems and lemmas used in the later proofs, such as Hadamard's inequality and Cramer's rule. Next, we establish the sufficient condition on the augmentation vector $\boldsymbol{v}$ obtained from the (2) step to ensure that a left kernel basis $\boldsymbol{C}$ of $\boldsymbol{v}$ has the property that $\boldsymbol{CA}$ is a basis for the given integer matrix $\boldsymbol{A}$.

In Chapter 3, we will adapt from the literature some computational tools to deterministically compute an augmentation vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ for input $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ such that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$. We will cover deterministic algorithms for system solving, 2-massager computation, LSP decomposition, and kernel basis problems. The deterministic algorithms are used in the sub problems of finding a maximal nonsingular submatrix of a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ and computing a kernel basis of a nullity-one matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$. We will then combine the two subproblems to produce the augmentation vector $\boldsymbol{v}$.

In the next two chapters, we will develop two different methods for computing a left kernel basis of a vector with nonzero entries. These two methods differ in bit operations and the output left kernel bases from them also have different properties. Let $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ be a vector with nonzero entries. In Chapter 4, we will introduce a divide-and-conquer algorithm that computes a sparse left kernel basis of any vectors with nonzero entries. The output left kernel basis of input $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ from this method is sparse and its bitlength is bounded by $2\log ||\boldsymbol{v}||$. We will demonstrate the tightness and sparsity of the output basis produced by the algorithm.

In Chapter 5, we will show a different approach to compute a left kernel basis of any vectors with nonzero entries via Hermite norm form. Although the output left kernel basis of $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ produced by this approach is not guaranteed to be sparse as in Chapter 4, its bitlength is better: only bounded by $\log ||\boldsymbol{v}||$.

In Chapter 6, we will present our main contribution that computes a lattice basis generator $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ for a given full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ such that $\boldsymbol{CA}$ is a basis for $\boldsymbol{A}$ by using the two aforementioned different methods to compute a kernel basis of a vector: sparse kernel basis from Chapter 4 and kernel basis of a vector via Hermite normal form from Chapter 5.

In Chapter 7, we will discuss our further exploration by extending the algorithm to the general case that the input matrix with full column rank can have more than one extra row. That is, given a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-k)}$, we can deterministically compute a lattice basis generator $\boldsymbol{C} \in \mathbb{Z}^{(n-k)\times n}$ such that $\boldsymbol{CA} \in \mathbb{Z}^{(n-k)\times(n-k)}$ is a basis for $\boldsymbol{A}$. We will discuss two approaches. The first approach is by iteratively applying our lattice generator algorithm derived in Chapter 6. The second approach is by iteratively applying the augmentation vector algorithm derived in Chapter 3 and then computing a kernel of a matrix.

1.1 Cost model

In this section, we give definitions and assumptions regarding the cost model that we will follow in this thesis.

We first recall that $\omega \in \mathbb{R}$ is a feasible matrix multiplication exponent such that two $n \times n$ matrices over field K can be multiplied with $O(n^\omega)$ operations in K. In this thesis, we assume $\omega > 2$.

Following von zur Gathen and Gerhard [2013, Section 8.3], cost estimates are given using a function $\mathsf{M}(d)$ that bounds the number of bit operations required to multiply two integers bounded in magnitude by 2^d. The recent algorithm of [Harvey and van der Hoeven, 2021] allows for $\mathsf{M}(d) \in O(d \log d)$.

Moreover, we assume that M holds the following assumptions:

(1) Superlinearity: $\mathsf{M}(a) + \mathsf{M}(b) \leq \mathsf{M}(a + b)$ for $a, b \in \mathbb{Z}_{\geq 1}$.
(2) Subquadratic: $\mathsf{M}(d) \in O(d^2)$.
(3) Submultiplicativity: $\mathsf{M}(ab) \leq \mathsf{M}(a)\mathsf{M}(b)$ for $a, b \in \mathbb{Z}_{\geq 1}$.
(4) Fast matrix implies fast integer: $\mathsf{M}(d) \in O(d^{\omega-1})$.

The first two assumptions are from von zur Gathen and Gerhard [2013, Definition 8.26]. The last two assumptions are commonly used such as in Storjohann [2005], Birmpilis et al. [2020]. Assumption (4) simply stipulates that if fast matrix multiplication techniques are used, then fast integer multiplication techniques should also be used.

In addition, following Storjohann [2000, Section 1.2], we use $\mathsf{B}(d)$ to bound the cost of integer gcd-related computations with two integers bounded in magnitude by 2^d such as the extended euclidean algorithm. This means that $\mathsf{B}(d) = O(\mathsf{M}(d) \log d)$ if pseudo-linear integer multiplication is used, or $\mathsf{B}(d) \in O(\mathsf{M}(d))$ if $\mathsf{M}(d) \in \Omega(d^{1+\epsilon})$ for some $\epsilon > 0$. The four assumptions given above for M also apply to B.

The motivation behind our cost model is that almost all of our algorithms are reduced to a number of matrix multiplications with integer matrices that have sizes similar to the size of the input matrix in the overall problem. For example, two $n \times n$ integer matrices with entries bounded by 2^d can be multiplied in $O(n^\omega \, \mathsf{M}(\log n + d))$ bit operations.

The following lemma follows from the assumptions above M and B cost functions. The lemma will be used to simplify the cost estimate throughout the thesis.

Lemma 9. $\mathsf{B}(nd) \in O(n^{\omega-1} \, \mathsf{B}(d))$.

Proof. Since B is submultiplicative, we have $\mathsf{B}(nd) \leq \mathsf{B}(n)\mathsf{B}(d)$. By the assumption that fast matrix multiplication implies fast integer multiplication, we have $\mathsf{B}(n) \in O(n^{\omega-1})$. It now follows that $\mathsf{B}(nd) \in O(n^{\omega-1} \, \mathsf{B}(d))$. $\square$

Chapter 2

Mathematical results

This chapter has three subsections. The first two subsections give background. Section 2.1 adapts from the literature some basic concepts related to lattices, primitive matrices, and left kernel basis. Section 2.2 recalls two classical theorems: Hadamard's inequality and Cramer's rule. Hadamard's inequality is useful for determining a bound on the determinant of a matrix. Cramer's rule can be used to relate the solution of a linear system equation to the determinant of matrices. We will use these two theorems to give the bound on the outputted kernel basis of an input matrix with nullity one in our established algorithm. So the outputted kernel basis can be applied to the 2-massager system solving algorithm from [Birmpilis et al., 2019, Section 11].

Section 2.3 gives a new result. We establish, via a more general case, the sufficient condition on the augmentation vector. Given a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ is an augmentation vector for matrix $\boldsymbol{A}$ if $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$. We show that if $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ is an augmentation vector for a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, then a left kernel basis $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ of $\boldsymbol{v}$ has the property that $\boldsymbol{C}\boldsymbol{A}$ is a basis for $\boldsymbol{A}$.

In the next Chapter, we present an algorithm to compute an augmentation vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ for a given full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ efficiently.

2.1 Background on integer lattice

We begin with the definition of a left nullspace basis of a matrix over the field of rational numbers $\mathbb{Q}$.

Definition 10 (Left nullspace basis of a rational matrix)**.** *A* left nullspace basis *of an* $\boldsymbol{A} \in \mathbb{Q}^{n\times m}$ *of rank* r *is a matrix* $\boldsymbol{N} \in \mathbb{Q}^{(n-r)\times n}$ *such that*

(i) $\boldsymbol{N}$ *has rank* $n - r$*, and*

(ii) $\boldsymbol{N}\boldsymbol{A}$ *is the* $(n-r)\times m$ *zero matrix.*

Property (ii) of Definition 10 is equivalent to the set of all $\mathbb{Q}$-linear combinations of the rows of $\boldsymbol{N}$ generating $\{\boldsymbol{v} \in \mathbb{Q}^{1\times n} \mid \boldsymbol{v}\boldsymbol{A} = 0\}$.

One way to define a kernel basis is by using the property of a matrix being primitive. An important notion for us is that of a primitive integer matrix.

Definition 11 (Primitive matrix)**.** *A matrix* $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ *is* primitive *if one of the following conditions hold.*

(i) $(n > m)$ $\boldsymbol{A}$ *has full column rank* m *and* $\boldsymbol{I}_m$ *is a basis for* $\boldsymbol{A}$*.*

(ii) $(n < m)$ $\boldsymbol{A}$ *has full row rank* n *and* $\boldsymbol{I_n}$ *is a basis for* Transpose($\boldsymbol{A}$)*.*

(iii) $(n = m)$ $\boldsymbol{A}$ *is unimodular (that is,* $\det \boldsymbol{A} = \pm 1$*).*

Combining the above two definitions, we can formally define the notion of a left kernel basis of an integer matrix.

Definition 12 (Left kernel basis of an integer matrix)**.** *A* left kernel basis *of an* $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ *with rank of* r *is a matrix* $\boldsymbol{K} \in \mathbb{Z}^{(n-r)\times n}$ *such that*

(i) $\boldsymbol{K}$ *has rank* $n - r$*,*

(ii) $\boldsymbol{K}\boldsymbol{A}$ *is the* $(n-r)\times m$ *zero matrix, and*

(iii) $\boldsymbol{K}$ *is primitive.*

We remark that a matrix $\boldsymbol{K}$ that satisfies only properties (i) and (ii) of Definition 12 is a nullspace basis of $\boldsymbol{A}$ over $\mathbb{Q}$: property (ii) ensures that the set of all $\mathbb{Q}$-linear combinations of the rows of $\boldsymbol{K}$ generates $\{\boldsymbol{v} \in \mathbb{Q}^{1\times n} \mid \boldsymbol{v}\boldsymbol{A} = \boldsymbol{0}_{1\times m}\}$. The additional property (iii) in Definition 12 ensures that $\boldsymbol{K}$ satisfies $\mathcal{L}(\boldsymbol{K}) = \{\boldsymbol{v} \in \mathbb{Z}^{1\times n} \mid \boldsymbol{v}\boldsymbol{A} = \boldsymbol{0}_{1\times m}\}$.

The following example illustrates the difference between a left kernel over $\mathbb{Q}$ and a left kernel basis over $\mathbb{Z}$.

Example 13. *Let*

$$\boldsymbol{A} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \in \mathbb{Z}^{3\times 1}.$$

The matrix

$$\boldsymbol{N} = \begin{bmatrix} 3 & 0 & -1 \\ 2 & 11 & -8 \end{bmatrix} \in \mathbb{Z}^{2\times 3}.$$

is a left kernel of $\boldsymbol{A}$ *over* $\mathbb{Q}$ *but not a left kernel basis of* $\boldsymbol{A}$ *over* $\mathbb{Z}$*. In particular, for* $\boldsymbol{u} = \begin{bmatrix} 1 & 7 & -5 \end{bmatrix} \in \mathbb{Z}^{1\times 3}$*, we have* $\boldsymbol{u}\boldsymbol{A} = 0_{1\times 3}$ *but* $\boldsymbol{u} \notin \mathcal{L}(\boldsymbol{N})$*. Note that*

$$\boldsymbol{u} = \begin{bmatrix} -\frac{1}{11} & \frac{7}{11} \end{bmatrix} \boldsymbol{N}.$$

The matrix

$$\boldsymbol{K} = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 3 & -2 \end{bmatrix} \in \mathbb{Z}^{2\times 3}$$

is a left kernel of $\boldsymbol{A}$ *and is also primitive. Note that*

$$\boldsymbol{u} = \begin{bmatrix} 1 & 2 \end{bmatrix} \boldsymbol{K}.$$

Lemma 14. *Let* $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ *have full column rank. If* $\boldsymbol{B} \in \mathbb{Z}^{m\times m}$ *is a basis for* $\boldsymbol{A}$*, then* $\boldsymbol{A}\boldsymbol{B}^{-1}$ *is integral and primitive.*

Proof. Since $\boldsymbol{B}$ is a basis for $\boldsymbol{A}$, $\boldsymbol{B}$ is nonsingular and we have

$$\boldsymbol{U}\boldsymbol{A} = \begin{bmatrix} \boldsymbol{B} \\ \hline \\ \end{bmatrix} \tag{2.1}$$

for some unimodular matrix $\boldsymbol{U} \in \mathbb{Z}^{n\times n}$. Postmultiplying both sides of (2.1) by $\boldsymbol{B}^{-1}$ gives

$$\boldsymbol{U}\boldsymbol{A}\boldsymbol{B}^{-1} = \begin{bmatrix} \boldsymbol{I}_m \\ \hline \\ \end{bmatrix}. \tag{2.2}$$

Premultiplying both sides of equation (2.2) by $\boldsymbol{U}^{-1}$ gives

$$\boldsymbol{A}\boldsymbol{B}^{-1} = \boldsymbol{U}^{-1} \left[\begin{array}{c} \boldsymbol{I}_m \\ \hline \\ \\ \end{array} \right]. \tag{2.3}$$

Since $\boldsymbol{U}$ is unimodular, $\boldsymbol{U}^{-1}$ is integral. The right side of equation (2.3) is integral. Hence, $\boldsymbol{A}\boldsymbol{B}^{-1}$ is integral. It follows from equation (2.2) that $\boldsymbol{A}\boldsymbol{B}^{-1}$ is primitive. □

The next lemma was given by Storjohann [2000, Lemma 5.9] but without proof. We offer a proof here.

Lemma 15 (Pasting lemma)**.** *Let $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ of rank r be given. If $\boldsymbol{C} \in \mathbb{Z}^{r\times n}$ is such that $\boldsymbol{C}\boldsymbol{A}$ is a basis for $\boldsymbol{A}$, and $\boldsymbol{K} \in \mathbb{Z}^{(n-r)\times n}$ is a left kernel basis of $\boldsymbol{A}$, then*

$$\left[\begin{array}{c} \boldsymbol{C} \\ \hline \boldsymbol{K} \end{array} \right] \in \mathbb{Z}^{n\times n} \tag{2.4}$$

is unimodular.

Proof. Since $\boldsymbol{K}$ is primitive, it follows from Definition 11.(ii) that there exists a unimodular matrix $\boldsymbol{U}$ such that $\boldsymbol{K}\,\boldsymbol{U} = \left[\begin{array}{c|c} & \boldsymbol{I}_{n-r} \end{array} \right]$. Let

$$\left[\begin{array}{c} \boldsymbol{C} \\ \boldsymbol{K} \end{array} \right] \boldsymbol{U} = \left[\begin{array}{c|c} \boldsymbol{V} & * \\ \hline & \boldsymbol{I}_{n-r} \end{array} \right] \tag{2.5}$$

and

$$\boldsymbol{U}^{-1}\boldsymbol{A} = \left[\begin{array}{c} \boldsymbol{A}_1 \\ \hline \boldsymbol{A}_2 \end{array} \right], \tag{2.6}$$

where $\boldsymbol{A}_1 \in \mathbb{Z}^{r\times m}$. Multiplying the right hand side of (2.5) by that of (2.6) gives

$$\left[\begin{array}{c|c} \boldsymbol{V} & * \\ \hline & \boldsymbol{I}_{n-r} \end{array} \right] \left[\begin{array}{c} \boldsymbol{A}_1 \\ \hline \boldsymbol{A}_2 \end{array} \right] = \left[\begin{array}{c} \boldsymbol{C}\boldsymbol{A} \\ \hline \\ \end{array} \right]. \tag{2.7}$$

We conclude from (2.7) that $\boldsymbol{A}_2$ is the zero matrix and then from (2.6) that $\boldsymbol{A}_1$ is a basis for $\boldsymbol{A}$. Since $\boldsymbol{C}\boldsymbol{A}$ is also a basis for $\boldsymbol{A}$, the matrix $\boldsymbol{V}$ must be unimodular. The result now follows from (2.5). □

2.2 Hadamard's inequality and Cramer's rule

The following two theorems are classical.

Theorem 16 (Hadamard's inequality)**.** *Let* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$*, with row vectors* $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n$*, and* $B \in \mathbb{R}$ *such that all entries of* $\boldsymbol{A}$ *are at most* B *in absolute value. Then*

$$|\det \boldsymbol{A}| \leq ||\boldsymbol{f}_1|| \ldots ||\boldsymbol{f}_n|| \leq n^{n/2} B^n$$

The proof of Hadamard's inequality is covered in [von zur Gathen and Gerhard, 2013, Theorem 16.6].

Theorem 17 (Cramer's rule)**.** *Let* $\boldsymbol{A} \in \mathbb{F}^{n\times n}$ *be nonsingular and* $\boldsymbol{b} \in \mathbb{F}^n$*. Then the coefficients of the unique solution* $\boldsymbol{y} = (y_1, \ldots, y_n)^T \in \mathbb{F}^n$ *of the linear system* $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$ *are given by*

$$y_i = \frac{\det \boldsymbol{A}_i}{\det \boldsymbol{A}},$$

where $\boldsymbol{A}_i \in \mathbb{F}^{n\times n}$ *is the matrix* $\boldsymbol{A}$ *with the ith column replaced by* $\boldsymbol{b}$*.*

The proof of Cramer's rule is covered in [von zur Gathen and Gerhard, 2013, Theorem 25.6].

2.3 Lattice generator via kernel basis

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

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

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

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

Theorem 18. *Let* $\boldsymbol{A} = \left[\begin{array}{c|c} \boldsymbol{A}_1 & \boldsymbol{A}_2 \end{array}\right] \in \mathbb{Z}^{n\times n}$ *be nonsingular, where* $\boldsymbol{A}_1 \in \mathbb{Z}^{n\times m_1}$ *and* $\boldsymbol{A}_2 \in \mathbb{Z}^{n\times m_2}$*. For any* $\boldsymbol{C}_1 \in \mathbb{Z}^{m_1\times n}$ *and* $\boldsymbol{C}_2 \in \mathbb{Z}^{m_2\times n}$*, we have* *(a)* $\left[\begin{array}{c} \boldsymbol{C}_1 \\ \hline \boldsymbol{C}_2 \end{array}\right] \left[\begin{array}{c|c} \boldsymbol{A}_1 & \boldsymbol{A}_2 \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{C}_1\boldsymbol{A}_1 & \\ \hline & \boldsymbol{C}_2\boldsymbol{A}_2 \end{array}\right] \in \mathbb{Z}^{n\times n}$, *with*

(b) $\boldsymbol{C}_1\boldsymbol{A}_1$ *a basis for* $\boldsymbol{A}_1$, *and*

(c) $\boldsymbol{C}_2\boldsymbol{A}_2$ *a basis for* $\boldsymbol{A}_2$

if and only if

(i) $\left[\begin{array}{c} \boldsymbol{C}_1 \\ \hline \boldsymbol{C}_2 \end{array}\right] \in \mathbb{Z}^{n\times n}$ *is unimodular,*

(ii) $\boldsymbol{C}_1$ *is a kernel basis for* $\boldsymbol{A}_2$, *and*

(iii) $\boldsymbol{C}_2$ *is a kernel basis for* $\boldsymbol{A}_1$.

Proof. **(Only if:)** Assume that (a), (b) and (c) hold. Since $\boldsymbol{A} = \left[\begin{array}{c|c} \boldsymbol{A}_1 & \boldsymbol{A}_2 \end{array}\right]$ is nonsingular, $\boldsymbol{A}_i$ has full column rank ($i = 1, 2$). From (b) and (c), we have that $\boldsymbol{C}_i\boldsymbol{A}_i \in \mathbb{Z}^{m_i\times m_i}$ also has full column rank ($i = 1, 2$). It follows that $\boldsymbol{C}_i\boldsymbol{A}_i$ is nonsingular ($i = 1, 2$).

Postmultiplying both sides of the equation in (a) by

$$\left[\begin{array}{c|c} (\boldsymbol{C}_1\boldsymbol{A}_1)^{-1} & \\ \hline & (\boldsymbol{C}_2\boldsymbol{A}_2)^{-1} \end{array}\right]$$

gives

$$\left[\begin{array}{c} \boldsymbol{C}_1 \\ \hline \boldsymbol{C}_2 \end{array}\right] \left[\begin{array}{c|c} \boldsymbol{A}_1(\boldsymbol{C}_1\boldsymbol{A}_1)^{-1} & \boldsymbol{A}_2(\boldsymbol{C}_2\boldsymbol{A}_2)^{-1} \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{I}_{m_1} & \\ \hline & \boldsymbol{I}_{m_2} \end{array}\right]. \tag{2.8}$$

Since $(\boldsymbol{C}_i\boldsymbol{A}_i)$ is a basis for $\boldsymbol{A}_i$ ($i = 1, 2$), it follows from Lemma 14 that

$$\boldsymbol{U} := \left[\begin{array}{c|c} \boldsymbol{A}_1(\boldsymbol{C}_1\boldsymbol{A}_1)^{-1} & \boldsymbol{A}_2(\boldsymbol{C}_2\boldsymbol{A}_2)^{-1} \end{array}\right]$$

is an integer matrix. Therefore,

$$\det\left(\left[\begin{array}{c} \boldsymbol{C}_1 \\ \hline \boldsymbol{C}_2 \end{array}\right]\right) = \pm 1$$

and property (i) holds. It remains to show that (ii) and (iii) hold. Since $\boldsymbol{U}$ is an integer matrix, it follows from (2.8) that $\boldsymbol{C}_1$ satisfies the properties required by Definition 12 to be a left kernel of $\boldsymbol{A}_2$. In particular, $\boldsymbol{C}_1$ is primitive and $\boldsymbol{C}_1\boldsymbol{A}_2 = \boldsymbol{0}_{m_1\times m_2}$. Similarly, $\boldsymbol{C}_2$ is a left kernel of $\boldsymbol{A}_1$.

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

$$\mathcal{L}\left(\left[\begin{array}{c|c} \boldsymbol{A}_1 & \boldsymbol{A}_2 \end{array}\right]\right) = \mathcal{L}\left(\left[\begin{array}{c|c} \boldsymbol{C}_1\boldsymbol{A}_1 & \\ \hline & \boldsymbol{C}_2\boldsymbol{A}_2 \end{array}\right]\right).$$

Properties (b) and (c) now follow by noting that $\boldsymbol{A}_i$ and $\boldsymbol{C}_i\boldsymbol{A}_i$ have the same column dimension ($i = 1, 2$). □

The following corollary identifies a particular class of matrices for which Theorem 18 applies.

Corollary 19. *Let $\boldsymbol{A} = \left[\begin{array}{c|c} \boldsymbol{A}_1 & \boldsymbol{A}_2 \end{array}\right] \in \mathbb{Z}^{n\times n}$ be nonsingular, where $\boldsymbol{A}_1 \in \mathbb{Z}^{n\times m_1}$ and $\boldsymbol{A}_2 \in \mathbb{Z}^{n\times m_2}$. If $\mathcal{L}(\boldsymbol{A})$ contains the last m_2 rows of $\boldsymbol{I}_n$, then any left kernel basis $\boldsymbol{C} \in \mathbb{Z}^{m_1\times n}$ of $\boldsymbol{A}_2$ has the property that $\boldsymbol{C}\boldsymbol{A}_1$ is a basis for $\boldsymbol{A}_1$.*

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

$$\boldsymbol{U}\boldsymbol{A} = \left[\begin{array}{c|c} * & \\ \hline & \boldsymbol{I}_{m_2} \end{array}\right].$$

Decompose $\boldsymbol{U}$ as

$$\boldsymbol{U} = \left[\begin{array}{c} \boldsymbol{C}_1 \\ \hline \boldsymbol{C}_2 \end{array}\right],$$

where $\boldsymbol{C}_1 \in \mathbb{Z}^{m_1\times n}$ and $\boldsymbol{C}_2 \in \mathbb{Z}^{m_2\times n}$. Then the equation in the property (a) of Theorem 18 holds, and from the unimodularity of $\boldsymbol{U}$, it follows that $\boldsymbol{C}_1$ is a left kernel basis for $\boldsymbol{A}_2$, and $\boldsymbol{C}_2$ is a left kernel basis for $\boldsymbol{A}_1$. We are thus exactly in the situation of Theorem 18, and can conclude that $\boldsymbol{C}_1\boldsymbol{A}_1$ is a basis for $\boldsymbol{A}_1$. Since any two left kernel bases of $\boldsymbol{A}_2$ are left equivalent, we can replace $\boldsymbol{C}_1$ with $\boldsymbol{C}$ in $\boldsymbol{U}$ and the matrix remains unimodular. □

Chapter 3

Computational results

In this chapter, we present some computational results that we will need throughout the rest of the thesis in order to deterministically compute an augmentation vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ for input matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ such that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$.

Section 3.1 recalls some essential computational tools that we will need throughout the rest of the thesis in order to establish the complexity bounds of our algorithms. Section 3.2 shows how to find a maximal nonsingular submatrix of any full column rank matrix. Section 3.3 then shows how to compute a left kernel basis of a full column rank input matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ after finding a maximal nonsingular submatrix of $\boldsymbol{A}$. Section 3.4 gives an algorithm to find an augmentation vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ such that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$ with entries not much larger than those of input matrix $\boldsymbol{A}$.

3.1 Computational tools

In this section, we present some essential computational tools related to deterministic system solving.

The main computational tool we need is an algorithm for nonsingular linear system solving. A deterministic algorithm has recently been given by Birmpilis et al. [2019, Section 11], but was analyzed under a more restrictive cost model than we are using in this thesis. We restate the result here.

Theorem 20 (System solving)**.** *There exists an algorithm that takes as input a nonsingular matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times n}$ and a vector $\boldsymbol{b} \in \mathbb{Z}^{n\times 1}$, and returns as output $\boldsymbol{A}^{-1}\boldsymbol{b} \in \mathbb{Q}^{n\times 1}$, together* *with the minimal* $e \in \mathbb{Z}_{>0}$ *such that* $e\boldsymbol{A}^{-1}\boldsymbol{b}$ *is integral. Let* $d = \log n + \log ||\boldsymbol{A}||$. *If* $\log ||\boldsymbol{b}|| \in O(nd)$, *the running time of the algorithm is* $O(n^\omega \mathsf{B}(d)(\log n)^2)$ *bit operations.*

Proof. [Birmpilis et al., 2019, Section 11] gives an algorithm that computes an integer vector $\boldsymbol{x} \in \mathbb{Z}^{n\times 1}$ together with an $f \in \mathbb{Z}_{\geq 0}$ such that $2^f \boldsymbol{A}^{-1}\boldsymbol{b}$ can be recovered from $\boldsymbol{x}$ using rational number reconstruction. The cost bound of [Birmpilis et al., 2019, Theorem 22] is

$$O(n^\omega \mathsf{M}(d)(\log n + \operatorname{loglog} ||\boldsymbol{A}||)(\log n)). \tag{3.1}$$

The cost model used in [Birmpilis et al., 2019] does not make the assumption that $\omega > 2$ and $\mathsf{M}(d) \in O(d^{\omega-1})$. If we do make these assumptions, and we replace the subroutine in [Birmpilis et al., 2019, Section 6] with the integer analogue of the algorithm supporting [Gupta et al., 2012, Theorem 7], the $\operatorname{loglog} ||\boldsymbol{A}||$ term in (3.1) can be avoided, giving the cost bound $O(n^\omega \mathsf{M}(d)(\log n)^2)$ to recover $\boldsymbol{x}$.

To recover $\boldsymbol{A}^{-1}\boldsymbol{b}$ from $\boldsymbol{x}$ requires rational number reconstruction. By [von zur Gathen and Gerhard, 2013, Section 5.10], the cost of computing e and $e\boldsymbol{A}^{-1}\boldsymbol{b}$ is $O(n\,\mathsf{B}(nd))$ bit operations; this simplifies to $O(n^\omega \mathsf{B}(d))$ using the assumptions $\mathsf{B}(nd) \in O(\mathsf{B}(n)\mathsf{B}(d))$ and $\mathsf{B}(n) \in O(n^{\omega-1}\mathsf{B}(d))$. □

The algorithm supporting Theorem 20 (System solving) was based on computing a 2-massager for the input matrix. Although originally defined for nonsingular matrices, the notion of a 2-massager extends directly to matrices of full column rank.

Definition 21 (2-Smith form)**.** *Let* $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ *have full column rank* m. *The* 2-Smith form *of* $\boldsymbol{A}$ *is the matrix* $\operatorname{diag}(2^{e_1}, 2^{e_2}, ..., 2^{e_m})$ *with* 2^{e_i} *the largest power of two which divides the invariant factor* i *of the Smith form of* $\boldsymbol{A}$ *over* $\mathbb{Z}, 1 \leq i \leq m$.

Definition 22 (2-massager)**.** *Let* $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ *have full column rank* m. *A* 2-massager *for* $\boldsymbol{A}$ *is a triple of matrices* $(\boldsymbol{P}, \boldsymbol{S}, \boldsymbol{M})$ *from* $\mathbb{Z}^{m\times m}$ *such that,*

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

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

Example 23. *A 2-massager for the matrix*

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

is given by

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

The massaged matrix

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

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

Birmpilis et al. [2019, Section 10] give an algorithm to compute a 2-massager of a nonsingular matrix. The algorithm extends naturally to an $n \times m$ matrix of full column rank m.

Theorem 24 (2-massager computation)**.** *There exists an algorithm that takes as input a full column rank $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, and returns as output a reduced 2-massager for $\boldsymbol{A}$ together with the massaged matrix $\boldsymbol{B} := \boldsymbol{A}\boldsymbol{P}\boldsymbol{M}\boldsymbol{S}^{-1}$. The cost of the algorithm is $O(nm^{\omega-1}\,\mathsf{M}(d)(\log m)^2)$ bit operations, where $d = \log m + \log ||\boldsymbol{A}||$.*

Proof. Similar to the proof of Theorem 20 (System solving), if we replace the subroutine in [Birmpilis et al., 2019, Section 6] with the integer analogue of the algorithm supporting [Gupta et al., 2012, Theorem 7], then [Birmpilis et al., 2019, Theorem 21] establishes the theorem in the case $n = m$, that is, for a square nonsingular $m \times m$ input matrix, a

2-massager can be computed in

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

bit operations. The only required modifications now to handle the case of a full column rank rectangular $n \times m$ input matrix is to incorporate blocking into steps 1–4 of the algorithm supporting [Birmpilis et al., 2019, Theorem 19].

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

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

Steps 3 and 4 compute a triangular Smith form of a square matrix over the ring. In the rectangular case, these steps are accomplished using the integer analogue of [Gupta et al., 2012, Theorem 7] followed up by the triangularization of a matrix that, compared to the square case, has row dimension now bounded by n. The cost increases similarly as for step 2. $\square$

The LSP decomposition of Ibarra et al. [1982] will be used in finding a maximal nonsingular submatrix later.

Definition 25 (LSP decomposition)**.** *An* ***LSP*** *decomposition of a matrix* $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ *is a triple of matrices* $(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{P})$*, where* $\boldsymbol{L} \in \mathbb{Z}^{n\times n}$ *is a lower triangular matrix with* 1*'s on the main diagonal,* $\boldsymbol{S} \in \mathbb{Z}^{n\times m}$ *is semi-upper triangular (i.e. deleting the zero rows of* $\boldsymbol{S}$ *yields an upper triangular matrix whose entries on the main diagonal are nonzero), and* $\boldsymbol{P} \in \mathbb{Z}^{n\times n}$ *is a permutation matrix.*

The following example comes from Jeannerod et al. [2013].

Example 26.

$$\overset{\boldsymbol{AP}^T}{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}} = \overset{\boldsymbol{L}}{\begin{bmatrix} 1 & & & & & & \\ * & 1 & & & & & \\ * & & 1 & & & & \\ * & & & 1 & & & \\ * & & & * & 1 & & \\ * & & & * & * & 1 & \\ * & & & * & * & & 1 \end{bmatrix}} \overset{\boldsymbol{S}}{\begin{bmatrix} \bar{*}_1 & * & * & * & * \\ & & & & \\ & & & & \\ & \bar{*}_2 & * & * & * \\ & & \bar{*}_3 & * & * \\ & & & & \\ & & & & \end{bmatrix}}$$

The matrix $\boldsymbol{S}$ *reveals that the row rank profile of* $\boldsymbol{A}$ *is* $[1, 4, 5]$.

Recall that the row rank profile of a matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ of rank r is the lexicographically minimal list of row indices $[i_1, i_2, \ldots, i_r]$ such that the corresponding rows of $\boldsymbol{A}$ are linearly independent. The rank profile corresponds to the pivot entries in the column echelon form.

Example 27. *If* $\boldsymbol{A} \in \mathbb{Z}^{6\times 4}$ *has column echelon form*

$$\begin{bmatrix} 1 & & & \\ * & & & \\ * & 1 & & \\ * & * & 1 & \\ * & * & * & \\ * & * & * & 1 \end{bmatrix} \in \mathbb{Q}^{6\times 4},$$

then the row rank profile of $\boldsymbol{A}$ *is* $[1, 3, 4, 6]$.

The following theorem is due to Ibarra et al. [1982, Theorem 3.2].

Theorem 28 (LSP decomposition solver)**.** *There is an algorithm to find an* `LSP` *decomposition of any* $n \times m$ *matrix over field* K*, where* $m \leq n$*. The cost is* $O(m^{\omega-1}n)$ *field operation.*

Storjohann [2000, Section 6.2] provided an algorithm to construct a kernel basis for a matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ with $rank(\boldsymbol{A}) = r$ bounded by m. We will use this algorithm in the last section.

Theorem 29 (Kernel of matrices)**.** *There exists an algorithm that takes input as a non-zeros matrix* $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ *with* $rank(\boldsymbol{A}) = r$ *bounded by* m*, and returns the output as a kernel basis* $\boldsymbol{M} \in \mathbb{Z}^{(n-r)\times n}$ *for* $\boldsymbol{A}$*. The cost of the algorithm is* $O(nm^{\omega-1}\log(2n/m)\,\mathsf{B}(d))$ *word operations, where* $d = m\log(m/2) + m\log||\boldsymbol{A}||$*. The kernel basis* $\boldsymbol{M}$ *will satisfy* $\log||\boldsymbol{M}|| \leq (1+m)\log m + 2m\log||A||$.

One additional algorithm that we need is a fast solution to the vector extended gcd problem [Storjohann, 2000, Corollary 6.5].

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

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

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

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

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

Figure 3.1: Algorithm `VectorGcdex`

3.2 Finding a maximal nonsingular submatrix

In this section, we show how to deterministically find a maximal nonsingular submatrix of a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ by using Theorem 24 (2-massager computation) and Theorem 28 (LSP decomposition solver).

Theorem 30 (Maximal nonsingular submatrix)**.** *A permutation matrix $\boldsymbol{Q} \in \mathbb{Z}^{n\times n}$ for a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ such that*

$$\boldsymbol{Q}\boldsymbol{A} = \begin{bmatrix} \bar{\boldsymbol{A}} \\ \hline \vdots \end{bmatrix} \in \mathbb{Z}^{n\times m} \tag{3.2}$$

with $\bar{\boldsymbol{A}} \in \mathbb{Z}^{m\times m}$ nonsingular can be computed in $O(nm^{\omega-1}\,\mathsf{M}(d)(\log m)^2)$ bit operations, where $d = \log m + \log ||\boldsymbol{A}||$.

Proof. To find a suitable $\boldsymbol{Q}$, compute a 2-massager $(\boldsymbol{P}, \boldsymbol{M}, \boldsymbol{S})$ of $\boldsymbol{A}$ together with the massaged matrix $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{P}\boldsymbol{M}\boldsymbol{S}^{-1}$. The massaged matrix $\boldsymbol{B}$ has full column rank modulo 2. Working modulo 2, use the LSP decomposition of Ibarra et al. [1982, Theorem 3.2] to compute the row rank profile $[i_1, i_2, \ldots, i_n]$ of $\boldsymbol{B}$ over $\mathbb{Z}/(2)$. Since $\boldsymbol{P}\boldsymbol{M}\boldsymbol{S}^{-1}$ is nonsingular, the submatrix of $\boldsymbol{A}$ comprised of rows $i_1, i_2, \ldots, i_n$ is also nonsingular. Define $\boldsymbol{Q}$ accordingly. The claim cost bound follows from Theorem 24 (2-massager computation) and 28 (LSP decomposition). □

Remark 31. *Despite that we can find the maximal singular submatrix of $\boldsymbol{A}$ using the massaged matrix $\boldsymbol{B}$, the row rank profile between $\boldsymbol{A}$ and $\boldsymbol{B}$ may not be equivalent.*

For example, let $\boldsymbol{A} = \left[\begin{array}{cc} 2 & 1 \end{array}\right]^T$, we have

$$\boldsymbol{B} = \operatorname{Rem}(\boldsymbol{APMS}^{-1}, 2) = \operatorname{Rem}\left(\left[\begin{array}{c} 2 \\ 1 \end{array}\right], 2\right) = \left[\begin{array}{c} 0 \\ 1 \end{array}\right]$$

Therefore, the row rank profile of $\boldsymbol{B}$ is 2 while the row rank profile of $\boldsymbol{A}$ is 1.

3.3 Kernel basis of nullity one

In this section, we give a deterministic algorithm to compute a left kernel basis $\boldsymbol{w} \in \mathbb{Z}^{1\times n}$ of a full column rank input matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$.

In this special case, where the nullity of $\boldsymbol{A}$ is one, the kernel basis $\boldsymbol{w}$ is unique (up to sign) and can be computed using nonsingular linear system solving. If $\boldsymbol{Q} \in \mathbb{Z}^{n\times n}$ is a permutation such that

$$\boldsymbol{QA} = \left[\begin{array}{c} \bar{\boldsymbol{A}} \\ \hline \boldsymbol{a} \end{array}\right] \in \mathbb{Z}^{n\times(n-1)} \tag{3.3}$$

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

$$\left[\begin{array}{c|c} \boldsymbol{a}\bar{\boldsymbol{A}}^{-1} & -1 \end{array}\right] \in \mathbb{Q}^{1\times n}.$$

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

$$\boldsymbol{w} = \boldsymbol{Q}\left[\begin{array}{c|c} e\boldsymbol{a}\bar{\boldsymbol{A}}^{-1} & -e \end{array}\right] \in \mathbb{Z}^{1\times n} \tag{3.4}$$

is primitive and is thus a kernel basis of $\boldsymbol{A}$ (Definition 12).

Theorem 32 (Kernel of nullity one)**.** *A left kernel basis of $\boldsymbol{w} \in \mathbb{Z}^{1\times n}$ of a full column rank $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ can be computed in $O(n^\omega \mathsf{B}(d)(\log n)^2)$ bit operations, where $d = \log n + \log ||\boldsymbol{A}||$.*

Proof. Firstly, compute the permutation matrix $\boldsymbol{Q} \in \mathbb{Z}^{n\times n}$ for $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ and return a nonsingular submatrix $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(n-1)\times(n-1)}$ of $\boldsymbol{QA}$ as in (3.3). Then, compute the minimal $e \in \mathbb{Z}_{>0}$ such that $e\boldsymbol{a}\bar{\boldsymbol{A}}^{-1}$ is integral, and compute $e\boldsymbol{a}\bar{\boldsymbol{A}}^{-1}$ itself. In the end, return $\boldsymbol{w}$ as in (3.4). The claimed cost bound follows from Theorems 30 (Maximal singular submatrix) and 20 (System solving). □

Corollary 33. *The kernel basis produced by Theorem 32 (Kernel of nullity one) satisfies* $\log ||\boldsymbol{w}|| \in O(nd)$.

Proof. Let $\boldsymbol{b} := \boldsymbol{a}\bar{\boldsymbol{A}}^{-1}$. Post-multiply both sides by $\bar{\boldsymbol{A}}$, we obtain $\boldsymbol{b}\bar{\boldsymbol{A}} = \boldsymbol{a}$. By Theorem 17 (Cramer's rule), we have $|e| \leq \det\big(\bar{\boldsymbol{A}}\big)$ and

$$||e\boldsymbol{b}|| \leq ||\det\big(\bar{\boldsymbol{A}}\big)\boldsymbol{b}|| \leq max_i||\det \bar{\boldsymbol{A}}_i|| \tag{3.5}$$

where $\bar{\boldsymbol{A}}_i \in \mathbb{Z}^{(n-1)\times(n-1)}$ is the matrix $\bar{\boldsymbol{A}}$ with the ith row replaced by $\boldsymbol{a}$. Since $||\boldsymbol{a}||, ||\bar{\boldsymbol{A}}|| \leq ||\boldsymbol{A}||$, by Theorem 16 (Hadamard's inequality), we can compute

$$\det \bar{\boldsymbol{A}}_i \leq (n-1)^{n-1/2}||\boldsymbol{A}||^{n-1} \tag{3.6}$$

and

$$\det \bar{\boldsymbol{A}} \leq n^n||\boldsymbol{A}||^n \tag{3.7}$$

The bound now follows from (3.4-3.7). □

3.4 Augmentation vector

In this section, we give a deterministic algorithm to find an augmentation vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ such that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$ with entries not much larger than those of input matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$.

We start with the definition of Hermite normal form over $\mathbb{Z}$.

Definition 34 (Hermite normal form)**.** *A nonsingular matrix* $\boldsymbol{H} \in \mathbb{Z}^{n\times n}$ *is said to be in Hermite normal form if all the following conditions are satisfied:*

- $\boldsymbol{H}$ *is upper-triangular*
- $\boldsymbol{H}_{i,i} \geq 1$, *for* $1 \leq i \leq n$
- $0 \leq \boldsymbol{H}_{i,j} < \boldsymbol{H}_{j,j}$, *for all* $1 \leq i < j \leq n$

Example 35.

$$\begin{bmatrix} 2 & 0 & 2 & 4 \\ & 1 & 0 & 15 \\ & & 6 & 4 \\ & & & 16 \end{bmatrix} \in \mathbb{Z}^{4\times 4}$$

is in Hermite normal form.

Storjohann [2005, Section 13] describes a Las Vegas randomized algorithm `DetReduction` that takes as input a nonsingular integer input matrix and replaces the last column to obtain a new matrix whose lattice contains the last row of $\boldsymbol{I}$. In other words, the new matrix has the same Hermite form $\boldsymbol{H}$ as the original matrix except with the last column equal to that of the identity matrix of the same dimension.

$$\boldsymbol{H} = \left[\begin{array}{ccc|c} * & * & * & \\ & * & * & \\ & & * & \\ & & & 1 \end{array}\right]$$

Example 36. *Consider the following input matrix*

$$\left[\begin{array}{cccc|c} -66 & -65 & 20 & -90 & 30 \\ 55 & 5 & -7 & -21 & 62 \\ 68 & 66 & 16 & -56 & -79 \\ 13 & -41 & -62 & -50 & 28 \\ 26 & -36 & -34 & -8 & -71 \end{array}\right] \in \mathbb{Z}^{5\times 5}$$

with Hermite normal form

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 10 & 260246748 \\ & 1 & 0 & 2 & 292062707 \\ & & 1 & 7 & 244095302 \\ & & & 14 & 342954195 \\ & & & & 344319363 \end{array}\right].$$

Algorithm `DetReduction` *produces the matrix*

$$\left[\begin{array}{cccc|c} -66 & -65 & 20 & -90 & 3 \\ 55 & 5 & -7 & -21 & 46 \\ 68 & 66 & 16 & -56 & 79 \\ 13 & -41 & -62 & -50 & -15 \\ 26 & -36 & -34 & -8 & 2 \end{array}\right]$$

with Hermite normal form

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 10 & \\ & 1 & 0 & 2 & \\ & & 1 & 7 & \\ & & & 14 & \\ & & & & 1 \end{array}\right]$$

A detailed description of algorithm `DetReduction` was not offered in Storjohann [2005, Chapter 6.1]; instead, a reference was given to a description of the polynomial analogue of the problem. For completeness, we give the details here. In addition, Algorithm `DetReduction` exploits the fact that the input matrix is nonsingular. Here, we adapt the approach to obtain the following result. In the proof of the following result, we focus mainly on showing how to avoid the requirement that the input matrix be nonsingular, and how to avoid randomization by using the subroutines developed in previous sections.

Theorem 37 (Augmentation vector)**.** *There exists an algorithm that takes as input a full column rank integer matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, and returns as output an integer vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ such that*

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

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

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

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

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

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

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

Then

$$\left[\begin{array}{c|c} \quad\bar{\boldsymbol{w}}\quad & w_n \end{array}\right] \left[\begin{array}{c|c} & \\ \bar{\boldsymbol{A}} & \bar{\boldsymbol{b}} \\ & \\ \hline \boldsymbol{a} & b_n \end{array}\right] = \left[\begin{array}{c|c} \quad\quad & 1 \end{array}\right]. \tag{3.8}$$

It follows from (3.8) that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{b} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$, and thus $\boldsymbol{b}$ is candidate for our solution vector. However, the entries of $\boldsymbol{b}$ can be too large.

We now show how to adjust $\boldsymbol{b}$ to obtain a reduced solution $\boldsymbol{v}$. Postmultiplying both sides of equation (3.8) by a unimodular matrix

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

gives

$$\left[\begin{array}{c|c} \quad\bar{\boldsymbol{w}}\quad & w_n \end{array}\right] \left[\begin{array}{c|c} & \\ \bar{\boldsymbol{A}} & \bar{\boldsymbol{b}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{y}} \\ & \\ \hline \boldsymbol{a} & b_n - \boldsymbol{a}\bar{\boldsymbol{y}} \end{array}\right] = \left[\begin{array}{c|c} \quad\quad & 1 \end{array}\right]. \tag{3.9}$$

Let $\boldsymbol{y} \in \mathbb{Q}^{(n-1)\times 1}$ be the solution to the linear system $\bar{\boldsymbol{A}}\boldsymbol{y} = \bar{\boldsymbol{b}}$. Let $\bar{\boldsymbol{y}} \in \mathbb{Z}^{(n-1)\times n}$ be the integral part of $\boldsymbol{y}$, that is, $\bar{\boldsymbol{y}}$ is the unique integer vector such that $||\boldsymbol{y} - \bar{\boldsymbol{y}}|| < 1$. Then $||\bar{\boldsymbol{b}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{y}}|| \leq n||\boldsymbol{A}||$. We choose our solution vector to be $\boldsymbol{v} := \boldsymbol{b} - \boldsymbol{A}\bar{\boldsymbol{y}}$. It remains to show that the last entry $v_n = b_n - \boldsymbol{a}\bar{\boldsymbol{y}}$ of $\boldsymbol{v}$ has magnitude bounded by $n^2||\boldsymbol{A}||$. From equation (3.9), we have $\bar{\boldsymbol{w}}\,\bar{\boldsymbol{v}} + w_n\,v_n = 1$. It follows that

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

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

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

Corollary 33 gives the bound $\log ||\boldsymbol{w}|| \in O(nd)$. From the specification of Algorithm 1 (VectorGcdex), we have $||\boldsymbol{b}|| \leq ||\boldsymbol{w}||$, so $\log ||\boldsymbol{b}|| \in O(nd)$ also. The claimed running time bound now follows from Theorem 32 (Kernel of nullity one), the running time of Algorithm 1 (VectorGcdex), and Theorem 20 (System solving). □

Example 38. *The input matrix*

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

has left kernel

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

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

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

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

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

Our augmentation vector is given by

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

Chapter 4

Sparse kernel basis of a vector

In this chapter, we present a divide-and-conquer approach to compute a sparse left kernel basis of any non-zero vectors and demonstrate the properties of the output basis. The output basis is sparse, with the number of its nonzero entries bounded by $O(n \log n)$. We derive an explicit upper bound on the number of non-zero entries in the output basis, as well as an explicit bound on the magnitude of entries.

The following example illustrates the sparsity of an output basis produced by our algorithm with a 16×1 non-zero input vector.

Example 39. *Given an input vector $\boldsymbol{v} \in \mathbb{Z}^{16\times 1}$ with all non-zero entries, our algorithm will output a sparse kernel basis $\boldsymbol{K} \in \mathbb{Z}^{15\times 16}$ of $\boldsymbol{v}$ that has the shape as*

$$\boldsymbol{K} = \begin{bmatrix}
* & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & & & & & & & & \\
* & * & * & * & & & & & & & & & & & & \\
* & * & & & & & & & & & & & & & & \\
& & * & * & & & & & & & & & & & & \\
& & & & * & * & * & * & & & & & & & & \\
& & & & * & * & & & & & & & & & & \\
& & & & & & * & * & & & & & & & & \\
& & & & & & & & * & * & * & * & * & * & * & * \\
& & & & & & & & * & * & * & * & & & & \\
& & & & & & & & * & * & & & & & & \\
& & & & & & & & & & * & * & & & & \\
& & & & & & & & & & & & * & * & * & * \\
& & & & & & & & & & & & * & * & & \\
& & & & & & & & & & & & & & * & *
\end{bmatrix} \in \mathbb{Z}^{15\times 16}.$$

Figure 4.1: Example sparse kernel basis

The algorithm we give is closely based on a more general algorithm [Storjohann, 2000, Chapter 6.2] that computes a left kernel basis for an $n \times m$ integer matrix. The variation we give here for the special case $m = 1$ is simpler and allows us to obtain a tighter explicit bound on the number of nonzero entries in the kernel.

```
Algorithm 2: SparseKernelBasis($\boldsymbol{v}, n$)
Input  : $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ with all entries nonzero, $n \in \mathbb{Z}$
Output : $\boldsymbol{K} \in \mathbb{Z}^{(n-1)\times n}$, a left kernel basis of $\boldsymbol{v}$ satisfying
    • $||\boldsymbol{K}|| \le ||\boldsymbol{v}||^2$, and
    • the number of nonzero entries in $\boldsymbol{K}$ is bounded by $n(1 + \log_2 n)$.
Runtime: $O(n\,\mathsf{B}(\log ||\boldsymbol{v}||) \log n)$ bit operations
1  if $n = 1$ then
2      return the $0 \times 1$ matrix
3  $n_1 := \lfloor n/2 \rfloor$
4  $n_2 := n - n_1$
5  Decompose $\boldsymbol{v} = \left[\begin{array}{c} \boldsymbol{v}_1 \\ \hline \boldsymbol{v}_2 \end{array}\right]$ where $\boldsymbol{v}_1 \in \mathbb{Z}^{n_1\times 1}$ and $\boldsymbol{v}_2 \in \mathbb{Z}^{n_2\times 1}$
6  $\boldsymbol{K}_1$ := SparseKernelBasis($\boldsymbol{v}_1, n_1$)
7  $\boldsymbol{K}_2$ := SparseKernelBasis($\boldsymbol{v}_2, n_2$)
8  $\boldsymbol{e}_1, g_1$ := VectorGcdex($\boldsymbol{v}_1, n_1$)
9  $\boldsymbol{e}_2, g_2$ := VectorGcdex($\boldsymbol{v}_2, n_2$)
10 $g := \gcd(g_1, g_2)$
11 return $\left[\begin{array}{c|c} -(g_2/g)\,\boldsymbol{e}_1 & (g_1/g)\,\boldsymbol{e}_2 \\ \hline \boldsymbol{K}_1 & \\ \hline & \boldsymbol{K}_2 \end{array}\right]$
```

Figure 4.2: Algorithm `SparseKernelBasis`

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

Lemma 40. *`SparseKernelBasis` outputs a kernel basis for* $\boldsymbol{v}$*.*

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

For the recursive case ($n > 1$), we have subproblems of size n_1 and n_2, with $n = n_1 + n_2$. By Lemma 15 (Pasting lemma), the matrix

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

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

$$\begin{array}{cccc} \boldsymbol{U} & \boldsymbol{v} & & \boldsymbol{c} \\ \left[\begin{array}{c|c} \boldsymbol{U}_1 & \\ \hline & \boldsymbol{U}_2 \end{array}\right] & \left[\begin{array}{c} \boldsymbol{v}_1 \\ \hline \boldsymbol{v}_2 \end{array}\right] & = & \left[\begin{array}{c} g_1 \\ \\ \hline g_2 \\ \\ \end{array}\right] \end{array} \in \mathbb{Z}^{n\times 1}, \tag{4.1}$$

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

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

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

that satisfies

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

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

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

Then $\boldsymbol{E}$ is unimodular with

$$\overset{\boldsymbol{E}}{\begin{bmatrix} *_1 & & & *_2 & & \\ & \ddots & & & & \\ & & 1 & & & \\ -g_2/g & & & g_1/g & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix}} \overset{\boldsymbol{c}}{\left[\begin{array}{c} g_1 \\ \\ \hline g_2 \\ \\ \end{array}\right]} = \overset{\bar{\boldsymbol{c}}}{\begin{bmatrix} g \\ \\ \\ \\ \\ \end{bmatrix}} \tag{4.2}$$

By (4.1) and (4.2), $(\boldsymbol{E}\boldsymbol{U})\boldsymbol{v} = \bar{\boldsymbol{c}}$. Since the last $n-1$ entries of $\bar{\boldsymbol{c}}$ are zeros, and $\boldsymbol{E}\boldsymbol{U}$ is unimodular, the last $n-1$ rows of $\boldsymbol{E}\boldsymbol{U}$ is a kernel basis of $\boldsymbol{v}$. The algorithm outputs the last $n-1$ rows of $\boldsymbol{E}\boldsymbol{U}$, except with row 1 and row n_1+1 interchanged. $\square$

Lemma 41. *The output $||\boldsymbol{K}||$ of* `SparseKernelBasis` *satisfies:*

(i) $||\boldsymbol{K}|| \leq ||\boldsymbol{v}||^2$*, and*

(ii) the number of nonzero entries in $\boldsymbol{K}$ is bounded by $n(1+\log_2 n)$.

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

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

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

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

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

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

From Sloane and Inc. [2020], we obtain that

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

Dropping the floor and ceiling and simplifying yields the claimed upper bound. $\square$

Lemma 42. *The running time of* `SparseKernelBasis` *is bounded by* $O(n\,\mathsf{B}(\log ||\boldsymbol{v}||) \log n)$ *bit operations.*

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

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

which solves to the running time bound stated in the lemma. $\square$

Theorem 43 (Sparse kernel basis of vectors)**.** *There exists an algorithm that takes as input a nonzero vector* $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$*, and returns as output a left kernel basis* $\boldsymbol{K} \in \mathbb{Z}^{(n-1)\times n}$ *for* $\boldsymbol{v}$*. The cost of the algorithm is* $O(n\,\mathsf{B}(\log ||\boldsymbol{v}||) \log n)$ *bit operations. The output will satisfy* $||\boldsymbol{K}|| \leq ||\boldsymbol{v}||^2$*. The number of nonzero entries in* $\boldsymbol{K}$ *will be bounded by* $n(1 + \log_2 n)$*.*

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

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

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

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

is a kernel basis for $\boldsymbol{v}$. The result now follows from Lemmas 40–42. $\square$

Chapter 5

Kernel basis of a vector via Hermite normal form

In Chapter 4 (Sparse kernel basis of a vector), we show how to compute a sparse left kernel basis of a non-zero column vector using a divide-and-conquer approach.

In this chapter, we provide a different approach to compute a left kernel basis of a non-zero vector via Hermite norm form. The output basis is no longer sparse, and the complexity of the method is a factor of n higher than the previous approach, but the magnitude bound on the output basis is smaller than that of the sparse kernel basis.

The idea of our approach is as follows. Decompose our input vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ as

$$\left[\begin{array}{c} v_1 \\ \hline \bar{\boldsymbol{v}} \end{array}\right] \in \mathbb{Z}^{n\times 1},$$

where $\bar{\boldsymbol{v}} \in \mathbb{Z}^{(n-1)\times 1}$ is the subvector comprised of the last $n-1$ entries of $\boldsymbol{v}$. Let

$$\boldsymbol{A} = \left[\begin{array}{c|c} v_1 & \\ \hline \bar{\boldsymbol{v}} & \boldsymbol{I}_{n-1} \end{array}\right] \in \mathbb{Z}^{n\times n}. \tag{5.1}$$

Then, we compute matrices $\boldsymbol{H}, \boldsymbol{U} \in \mathbb{Z}^{n\times n}$ such that $\boldsymbol{U}\boldsymbol{A} = \boldsymbol{H}$, where $\boldsymbol{H}$ is the Hermite normal form of $\boldsymbol{A}$ and $\boldsymbol{U} \in \mathbb{Z}^{n\times n}$ is a unimodular matrix.
We decompose $\boldsymbol{H}$ and $\boldsymbol{U}$ as

$$\boldsymbol{H} = \left[\begin{array}{c|c} * & * \\ \hline & \bar{\boldsymbol{H}} \end{array}\right] \in \mathbb{Z}^{n\times n} \tag{5.2}$$

and

$$\boldsymbol{U} = \left[\begin{array}{c|c} * & * \\ \hline \boldsymbol{K}_1 & \bar{\boldsymbol{H}} \end{array}\right] \in \mathbb{Z}^{n\times n} \tag{5.3}$$

where $\boldsymbol{K}_1 \in \mathbb{Z}^{(n-1)\times 1}$ and $\bar{\boldsymbol{H}} \in \mathbb{Z}^{(n-1)\times(n-1)}$.
Then $\boldsymbol{K} = \left[\begin{array}{c|c} \boldsymbol{K}_1 & \bar{\boldsymbol{H}} \end{array}\right] \in \mathbb{Z}^{(n-1)\times n}$ is a left kernel of $\boldsymbol{v}$.

Section 5.1 applies the Hermite normal form algorithm from Storjohann and Labahn [1996, Theorem 5] to compute the Hermite normal form (5.2) of the specially constructed matrix (5.1). Section 5.2 presents the algorithm to compute a left kernel basis of a non-zero vector by using the method described in the former section and shows the complexity of the algorithm and the magnitude bound of the output basis.

5.1 Special Hermite normal form

Let $\boldsymbol{v} = \begin{bmatrix} v_1 & v_2 & v_3 & \dots & v_n \end{bmatrix}^T \in \mathbb{Z}^{n\times 1}$ with $v_1 \neq 0$. In this section, we show how to compute the Hermite normal form of the structured matrix

$$\boldsymbol{A} = \left[\begin{array}{c|cccc} v_1 & & & & \\ \hline v_2 & 1 & & & \\ v_3 & & 1 & & \\ \vdots & & & \ddots & \\ v_n & & & & 1 \end{array}\right] \in \mathbb{Z}^{n\times n}. \tag{5.4}$$

Storjohann and Labahn [1996, Theorem 5] has given a deterministic algorithm to produce the Hermite normal form of a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ in $O(m^{\omega-1} n \log(2n/m) \mathsf{B}(m \log m ||\boldsymbol{A}||))$ bit operations. The term inside the B function is the bound of $\log \det \boldsymbol{A}$ by Theorem 16 (Hadamard's inequality).

We notice that, in our constructed matrix $\boldsymbol{A}$ (5.4), we have

$$\det \boldsymbol{A} = v_1 \leq ||\boldsymbol{v}|| \tag{5.5}$$

Since each step of computation in Storjohann and Labahn [1996, Theorem 5] has modulo $\det \boldsymbol{A}$, it follows from (5.5) that the term inside the B function for the given input (5.4) can be replaced by $\log ||\boldsymbol{v}||$. Here is the algorithm:

Theorem 44 (Special Hermite computation)**.** *There exists a deterministic algorithm that takes as input $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$, and returns as output a $\boldsymbol{H} \in \mathbb{Z}^{n\times n}$ such that $\boldsymbol{H}$ is the Hermite normal form of (5.4). The cost of the algorithm is $O(n^{\omega}\,\mathsf{B}(\log ||\boldsymbol{v}||))$ bit operations.*

We remark that the algorithm does not fully exploit the shape of our specially constructed matrix (5.4) but the cost of the Hermite form computation does not dominate the cost in our application.

5.2 Kernel basis of a vector via Hermite normal form

Let $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ where $v_1 \neq 0$ be given. In this section, we give an algorithm to compute a left kernel basis $\boldsymbol{K} \in \mathbb{Z}^{(n-1)\times n}$ for $\boldsymbol{v}$ by using the Hermite normal form algorithm in Theorem 44 (Special Hermite computation).

Let $\boldsymbol{v} = \begin{bmatrix} v_1 & v_2 & v_3 & \dots & v_n \end{bmatrix}^T \in \mathbb{Z}^{n\times 1}$ with $v_1 \neq 0$. We have

$$\boldsymbol{U} \left[\begin{array}{c|ccc} v_1 & & & \\ \hline v_2 & 1 & & \\ \vdots & & \ddots & \\ v_n & & & 1 \end{array}\right] = \overset{\boldsymbol{H}}{\begin{bmatrix} h_1 & * & \dots & * \\ & h_2 & \dots & * \\ & & \ddots & \vdots \\ & & & h_n \end{bmatrix}} \tag{5.6}$$

with unimodular matrix $\boldsymbol{U} \in \mathbb{Z}^{n\times n}$ and the right hand side in Hermite normal form. We can decompose $\boldsymbol{H}$ as:

$$\boldsymbol{H} = \left[\begin{array}{c|c} * & * \\ \hline & \bar{\boldsymbol{H}} \end{array}\right] \text{ where } \bar{\boldsymbol{H}} \in \mathbb{Z}^{(n-1)\times(n-1)}$$

Notice that $\boldsymbol{H}_{2..n,1} = \boldsymbol{0}_{(n-1)\times 1}$ and by equation (5.6), we have $\boldsymbol{U}_{2..n,2..n}\,\boldsymbol{v} = \boldsymbol{0}_{(n-1)\times 1}$. Since $\boldsymbol{U}$ is a unimodular matrix, $\boldsymbol{U}_{2..n,2..n}$ is a kernel basis of $\boldsymbol{v}$. Therefore, we don't need to compute all of $\boldsymbol{U}$. We only need to compute $\boldsymbol{U}_{2..n,2..n}$.

Decompose $\boldsymbol{U}$ as:

$$\boldsymbol{U} = \left[\begin{array}{c|c} * & * \\ \hline \boldsymbol{K}_1 & \boldsymbol{K}_2 \end{array}\right]$$

where $\boldsymbol{K}_1 \in \mathbb{Z}^{(n-1)\times 1}$ and $\boldsymbol{K}_2 \in \mathbb{Z}^{(n-1)\times(n-1)}$. In fact $\boldsymbol{K}_2 = \bar{\boldsymbol{H}}$ by equation (5.6), and we only need to construct the column $\boldsymbol{K}_1$.

```
Algorithm 3: KernelViaHermite($\boldsymbol{v}, n$)
   Input   : $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}, n \in \mathbb{Z}$ where $v_1 \neq 0$
   Output  : $\boldsymbol{K} \in \mathbb{Z}^{(n-1)\times n}$ is a kernel basis of $\boldsymbol{v}$ where $||\boldsymbol{K}|| \leq ||\boldsymbol{v}||$
1  $\bar{\boldsymbol{v}} := \boldsymbol{v}_{2,\ldots,n}$;
2  Compute Hermite basis: $\boldsymbol{H} := \left[\begin{array}{c|c} * & * \\ \hline & \bar{\boldsymbol{H}} \end{array}\right]$ of $\left[\begin{array}{c|c} \boldsymbol{v}_1 & \\ \hline \bar{\boldsymbol{v}} & \boldsymbol{I}_{n-1} \end{array}\right]$ where
     $\bar{\boldsymbol{H}} \in \mathbb{Z}^{(n-1)\times(n-1)}$;
3  return $\left[\begin{array}{c|c} -\bar{\boldsymbol{H}}\,\bar{v}/v_1 & \bar{\boldsymbol{H}} \end{array}\right]$
```

Figure 5.1: Algorithm `KernelViaHermite`

Lemma 45. *Algorithm* `KernelViaHermite` *is correct.*

Proof. Decompose $\boldsymbol{v}$ as

$$\left[\begin{array}{c} v_1 \\ \hline \bar{\boldsymbol{v}} \end{array}\right] \in \mathbb{Z}^{n\times 1},$$

where $\bar{\boldsymbol{v}} \in \mathbb{Z}^{(n-1)\times 1}$ is the subvector comprised of the last $n-1$ entries of $\boldsymbol{v}$. Let

$$\boldsymbol{A} = \left[\begin{array}{c|c} v_1 & \\ \hline \bar{\boldsymbol{v}} & \boldsymbol{I}_{n-1} \end{array}\right] \in \mathbb{Z}^{n\times n}. \tag{5.7}$$

Let $\boldsymbol{H} \in \mathbb{Z}^{n\times n}$ be the Hermite form of $\boldsymbol{A}$ and $\boldsymbol{U}\boldsymbol{A} = \boldsymbol{H}$, where $\boldsymbol{U} \in \mathbb{Z}^{n\times n}$ is a unimodular matrix. Decompose $\boldsymbol{H}$ and $\boldsymbol{U}$ as

$$\boldsymbol{H} = \left[\begin{array}{c|c} * & * \\ \hline & \bar{\boldsymbol{H}} \end{array}\right] \in \mathbb{Z}^{n\times n} \tag{5.8}$$

$$\boldsymbol{U} = \left[\begin{array}{c|c} * & * \\ \hline \boldsymbol{K}_1 & \bar{\boldsymbol{H}} \end{array}\right] \in \mathbb{Z}^{n\times n} \tag{5.9}$$

where $\boldsymbol{K}_1 \in \mathbb{Z}^{(n-1)\times 1}$ and $\bar{\boldsymbol{H}} \in \mathbb{Z}^{(n-1)\times(n-1)}$.

Then $\boldsymbol{K} = \left[\begin{array}{c|c} \boldsymbol{K}_1 & \bar{\boldsymbol{H}} \end{array}\right] \in \mathbb{Z}^{(n-1)\times n}$ is a left kernel of $\boldsymbol{v}$.

$\bar{\boldsymbol{H}}$ can be obtained from the corresponding positions of the Hermite matrix $\boldsymbol{H}$ as in equation (5.8). $\boldsymbol{K}_1$ can be obtained by using the formula:

$$\boldsymbol{K}_1 = \frac{-\bar{\boldsymbol{H}}\bar{\boldsymbol{v}}}{v_1} \tag{5.10}$$

Since $\boldsymbol{H}_{2..n,1} = \boldsymbol{0}_{(n-1)\times 1}$ and for $2 \leq i \leq n$, $\boldsymbol{H}_{i,1} = \boldsymbol{U}_{i,1}\boldsymbol{A}_{1,1} + \boldsymbol{U}_{i,2..n}\boldsymbol{A}_{2..n,1} = \boldsymbol{U}_{i,1}v_1 + \boldsymbol{U}_{i,2..n}\bar{\boldsymbol{v}}$, we have $\boldsymbol{K}_1 v_1 + \bar{\boldsymbol{H}}\,\bar{\boldsymbol{v}} = \boldsymbol{0}_{(n-1)\times 1}$.

□

Lemma 46. *Algorithm* `KernelViaHermite` *runs in time $O(n^\omega \mathsf{B}(\log ||\boldsymbol{v}||))$ bit operations.*

Proof. $\bar{\boldsymbol{H}}$ can be computed in $O(n^\omega \mathsf{B}(\log ||\boldsymbol{v}||))$ bit operations as in Theorem 44 (Special Hermite computation). To compute $\boldsymbol{K}_1$, we have a dot product between a $(n-1)\times(n-1)$ matrix and a $(n-1)\times 1$ vector. This gives us $O(n^2 \mathsf{M}(\log ||\boldsymbol{v}||))$ bit operations. Hence, the total cost is $O(n^\omega \mathsf{B}(\log ||\boldsymbol{v}||))$ bit operations. □

The remainder of this section is devoted to establishing the following result.

Theorem 47 (Magnitude bound)**.** *The output kernel $\boldsymbol{K}$ of Algorithm* `KernelViaHermite` *satisfies $||\boldsymbol{K}|| \leq ||\boldsymbol{v}||$.*

Since $\boldsymbol{K} = \left[\begin{array}{c|c} \boldsymbol{K}_1 & \bar{\boldsymbol{H}} \end{array}\right]$, we will firstly show $||\bar{\boldsymbol{H}}|| \leq ||\boldsymbol{v}||$ and then show $||\boldsymbol{K}_1|| \leq ||\boldsymbol{v}||$.

We will establish a sequence of lemmas to prove Theorem 47. We begin with the following remark.

Remark 48.

$$\begin{array}{ccc} \textit{Initial matrix} & & \textit{final matrix} \\ \begin{bmatrix} 1 & 3 & 6 & 0 & 2 \\ & 5 & 4 & 0 & 1 \\ & & 7 & 0 & 2 \\ & & & 1 & 1 \\ & & & & 3 \end{bmatrix} & \xrightarrow{\substack{\textit{swap col: 3,4} \\ \textit{swap row: 3,4} \\ \textit{swap col: 2,3} \\ \textit{swap row: 2,3}}} & \begin{bmatrix} 1 & 0 & 3 & 6 & 2 \\ & 1 & 0 & 0 & 1 \\ & & 5 & 4 & 1 \\ & & & 7 & 2 \\ & & & & 3 \end{bmatrix} \end{array} \tag{5.11}$$

Denote the number of diagonal entries of a Hermite form matrix $\boldsymbol{H} \in \mathbb{Z}^{n\times n}$ that are > 1 by k where $k \in [0, n]$. The number of 1 in the diagonal entries of $\boldsymbol{H}$ is $n - k$.
The Hermite form matrix $\boldsymbol{H}$ can be arranged by row and column permutations as in equation (5.11) such that the first $(n-k)$ diagonal entries are 1 and the last k diagonal entries are > 1.

Lemma 49. *We have $h_1 h_2 \cdots h_n = |v_1|$, where $h_1, \ldots, h_n$ are the diagonal entries of the Hermite form $\boldsymbol{H}$.*

Proof. Because $\boldsymbol{H}$ is unimodularly left equivalent to A, we have $\det \boldsymbol{H} = |\det \boldsymbol{A}|$. The lemma now follows from the fact in equation (5.7) that $\det \boldsymbol{A} = v_1$ and $\det \boldsymbol{H} = h_1 h_2 \cdots h_n$. □

With equation (5.8) and Lemma 49, we have $||\bar{\boldsymbol{H}}|| \leq |v_1| \leq ||\boldsymbol{v}||$. Therefore, $\bar{\boldsymbol{H}} \leq ||\boldsymbol{v}||$ is proved. Next, we will focus on proving $\boldsymbol{K}_1 \leq ||\boldsymbol{v}||$. We will firstly establish an inequality for $||\boldsymbol{K}_1||$:

Denote k_i for $2 \leq i \leq n$ as the *ith* entries of $\boldsymbol{K}_1$.

From equation (5.10), it follows that

$$k_i = -(\sum_{j=i}^{n} \boldsymbol{H}_{i,j} v_j)/v_1$$

$$|k_i| = |\sum_{j=i}^{n} \boldsymbol{H}_{i,j} v_j / v_1| = |\frac{1}{v_1} \sum_{j=i}^{n} \boldsymbol{H}_{i,j} v_j|$$

$$||\boldsymbol{K}_1|| = \frac{1}{|v_1|} \max_i |\sum_{j=i}^{n} \boldsymbol{H}_{i,j} v_j| \tag{5.12}$$

We now establish a bound for the right-hand side of inequality (5.10) using a sequence of lemmas.

Without loss of generality, we can assume that the first $(n-k)$ diagonal entries of the Hermite matrix $\boldsymbol{H}$ in equation (5.8) are 1 and the last k diagonal entries are > 1.

Lemma 50. *For $1 \leq i \leq n$, we have $h_i \leq \frac{|v_1|}{2^{k-1}}$ where $k \in [0, n]$ is the number of diagonal entries that are > 1.*

Proof. Since the first $n-k$ diagonal entries $= 1$, with Lemma 49,

$$|v_1| = \prod_{i=1}^{n} h_i = \left(\prod_{i=1}^{n-k} 1\right) \left(\prod_{i=n-k+1}^{n} h_i\right) = \prod_{i=n-k+1}^{n} h_i$$

Since $h_i > 1$ and h_i must be integer, we have $h_i \geq 2$. In addition, there are k such h_i which implies the inequality:

$$|v_1| \geq 2^{k-1} h_i$$

$$h_i \leq \frac{|v_1|}{2^{k-1}}$$

Therefore, $h_i \leq \frac{|v_1|}{2^{k-1}}$ and the statement is proved. □

Lemma 51. $\max_i(\sum_j \boldsymbol{H}_{i,j}) \leq |v_1|$

Proof. By the definition of Hermite form, we have $\boldsymbol{H}_{i,j} < \boldsymbol{H}_{i,i} = h_i$.
Without loss of generality, let $v_1 > 0$.

If $k = 0$ (equivalent to $v_1 = 1$), the Hermite matrix $\boldsymbol{H}$ would be an identity matrix. Therefore, $\max_i(\sum_j \boldsymbol{H}_{ij}) = 1 = v_1$. Next, we will prove the inequality when $v_1 > 1$ (equivalent to $1 \leq k \leq n$).

In the following, with Lemma 50, we will firstly establish an inequality

$$\sum_{j=1}^{n} \boldsymbol{H}_{i,j} \leq k \frac{v_1}{2^{k-1}} - k + 1 \tag{5.13}$$

by illustrating two cases in terms of the row i. Then we will show the right-hand side of the equation (5.13) $\leq v_1$.

As in Remark 48, the diagonal entries in the first $n-k$ rows are 1 and the diagonal entries in the last k rows are > 1. By the definition of the Row Hermite normal form, each diagonal entry has the largest value in its column. The two different cases can be defined as $h_i = 1$ and $h_i > 1$.

$$\left(\begin{array}{cccccc}
1 & & & * & * & * \\
 & 1 & & * & * & * \\
 & & \ddots & * & * & * \\
 & & & h_{n-k+1} & * & * \\
 & & & & \ddots & * \\
 & & & & & h_n
\end{array}\right)
\begin{array}{l}
\leftarrow \text{case 1: } h_i = 1 \\
 \\
 \\
\leftarrow \text{case 2: } h_i > 1 \\
 \\

\end{array}$$

Case 1: $h_i = 1$:

For $j \in [i+1, n-k]$, $\boldsymbol{H}_{i,j} = 0$ and for $j \in [n-k+1, n]$, $0 \le \boldsymbol{H}_{ij} \le h_j - 1$, we can establish the following equation:

$$\begin{aligned}
\sum_{j=1}^{n} \boldsymbol{H}_{i,j} &= 1 + \sum_{j=n-k+1}^{n} (\boldsymbol{H}_{i,j} - 1) \\
&\le 1 + \sum_{j=n-k+1}^{n} (h_j - 1) \\
&\le 1 + k\left(\frac{v_1}{2^{k-1}} - 1\right) \\
&\le k\frac{v_1}{2^{k-1}} - k + 1
\end{aligned}$$

Case 2: $h_i > 1$:

For $j \in [i+1, n]$, $\boldsymbol{H}_{i,j} \ge 0$, we can establish the following equation:

$$\begin{aligned}
\sum_{j=1}^{n} \boldsymbol{H}_{i,j} &= h_i + \sum_{j=i+1}^{n} \boldsymbol{H}_{i,j} \\
&\le h_i + \sum_{j=n-k+2}^{n} \boldsymbol{H}_{i,j} \\
&\le h_i + \sum_{j=n-k+2}^{n} (h_j - 1) \\
&\le \frac{v_1}{2^{k-1}} + (k-1)\left(\frac{v_1}{2^{k-1}} - 1\right) \\
&\le k\frac{v_1}{2^{k-1}} - k + 1
\end{aligned}$$

We now show that

$$k\frac{v_1}{2^{k-1}} - k + 1 \le v_1 \tag{5.14}$$

where $k \in [1, n]$. We will prove Inequality (5.14) by using derivative of $f(x) = x\frac{v_1}{2^{x-1}} - x + 1$ and show that $f(x)$ is a decreasing function. Then, we can derive that $f(x) \le f(1) = v_1$:

Let $f(x) = x(\frac{v_1}{2^{x-1}}) - x + 1$

$$\begin{aligned}\frac{df}{dx} &= \frac{d}{dx}(x)\left(\frac{v_1}{2^{x-1}}\right) + \frac{d}{dx}\left(\frac{v_1}{2^{x-1}}\right)x \\ &= \frac{v_1}{2^{x-1}} + (-2^{-x+1}v_1 \ln 2)x \\ &= \frac{v_1(1-\ln 2)x}{2^{x-1}} - 1\end{aligned}$$

Since $v_1 > 0$ and $(1 - \ln 2) < 0$, $\frac{df}{dx} < 0$ when $x \in [1, n]$ and $f(x)$ is decreasing when $x \in [1, n]$.

$$f(k) \leq f(1) \qquad \text{for } k \in [1, n]$$

$$\sum_{j=1}^{n} \boldsymbol{H}_{i,j} \leq f(k) \leq f(1) = v_1 - 1 + 1 = v_1$$

Hence, we have proved that $\max_i(\sum_j \boldsymbol{H}_{i,j}) \leq |v_1|$. □

Proof. (Of Theorem 47)

$\bar{\boldsymbol{H}} \leq ||\boldsymbol{v}||$ has been proved after Lemma 49.

We now prove $\boldsymbol{K}_1 \leq ||\boldsymbol{v}||$:

By Lemma 51 and Equation (5.10),

$$\begin{aligned}||\boldsymbol{K}_1|| &= \frac{1}{|v_1|} \max_i |\sum_{j=i}^{n} \boldsymbol{H}_{i,j} v_j| \\ &\leq \frac{1}{|v_1|} \max_i |\sum_{j=i}^{n} \boldsymbol{H}_{i,j}| \cdot ||\boldsymbol{v}|| \\ &\leq \frac{1}{|v_1|} |v_1| \cdot ||\boldsymbol{v}|| \\ &\leq ||\boldsymbol{v}||\end{aligned}$$

Since $||\boldsymbol{K}_1|| \leq ||\boldsymbol{v}||$, $||\bar{\boldsymbol{H}}|| \leq ||\boldsymbol{v}||$ and $\boldsymbol{K} = \left[\begin{array}{c|c} \boldsymbol{K}_1 & \bar{\boldsymbol{H}} \end{array}\right]$, $||\boldsymbol{K}|| \leq ||\boldsymbol{v}||$. □

Theorem 52 (Kernel basis of vectors via Hermite)**.** *There exists an algorithm that takes as input a nonzero vector $\boldsymbol{v} \in \mathbb{Z}^{n \times 1}$, and returns as output a left kernel basis $\boldsymbol{K} \in \mathbb{Z}^{(n-1)\times n}$ for $\boldsymbol{v}$. The cost of the algorithm is $O(n^\omega \mathsf{B}(\log ||\boldsymbol{v}||))$ bit operations. The output will satisfy $||\boldsymbol{K}|| \leq ||\boldsymbol{v}||$.*

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

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

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

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

is a kernel basis for $\boldsymbol{v}$. The result now follows from Lemmas 45–46 and Theorem 47 (Magnitude bound). $\square$

Chapter 6

Lattice generator

In this chapter, we introduce our two main deterministic algorithms to compute a lattice basis generator $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ for a given full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ using two different methods to compute a kernel basis of a vector: sparse kernel basis and kernel basis of a vector via Hermite normal form.

Both our algorithms share the same idea but are different in the last step that computing a left kernel basis of a vector. We now give an outline of our approaches to compute $\boldsymbol{C}$.

Our idea is to find a vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ for the input $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ such that the augmented matrix

$$\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right] \in \mathbb{Z}^{n\times n} \tag{6.1}$$

is nonsingular and $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$. Then we compute $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ to be a left kernel basis of $\boldsymbol{v}$ such that $\boldsymbol{C}\boldsymbol{A}$ is a basis for $\boldsymbol{A}$. In summary, the idea has three main computational steps:

1. Compute the left kernel basis $\boldsymbol{w} \in \mathbb{Z}^{1\times n}$ of $\boldsymbol{A}$ by Theorem 32 (Kernel of nullity one).
2. Compute an augmentation vector $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ from $\boldsymbol{A}$ and $\boldsymbol{w}$ by Theorem 37 (Augmentation vector).
3. Compute a left kernel basis $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ of $\boldsymbol{v}$ by Theorem 43 (Sparse kernel basis of vectors) or Theorem 52 (Kernel basis of vectors via Hermite).

Section 6.1 shows how to compute a lattice basis generator $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ for a given full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ by Theorem 43 (Sparse kernel basis of vectors) and also shows that $||\boldsymbol{C}|| \leq n^4||\boldsymbol{A}||^2$ and that $\boldsymbol{C}$ will have at most $n(1+\log_2 n)$ nonzero entries, independent of $||\boldsymbol{A}||$.

Section 6.2 shows how to compute a lattice basis generator $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ for a given full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ by Theorem 52 (Kernel basis of vectors via Hermite) and also show that $||\boldsymbol{C}|| \leq n^2||\boldsymbol{A}||$. Compared to the returned basis generator by the previous approach, the magnitude bound of $\boldsymbol{C}$ is smaller but is no longer guaranteed to be sparse.

6.1 Sparse lattice generator

In this section, we present a deterministic algorithm to compute a lattice basis generator $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ for a matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ via sparse kernel basis method.

Theorem 53 (Sparse Lattice generator)**.** *There exists an algorithm that takes as input a full column rank $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, and returns as output a $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ such that $\boldsymbol{CA}$ is a basis for $\boldsymbol{A}$. The output will satisfy $||\boldsymbol{C}|| \leq n^4||\boldsymbol{A}||^2$ and the number of nonzero entries in $\boldsymbol{C}$ will be bounded by $n(1+\log_2 n)$. The cost of the algorithm is $O(n^\omega \mathsf{B}(d)(\log n)^2)$ bit operations, where $d = \log n + \log ||\boldsymbol{A}||$.*

Proof. Use the algorithm supporting Theorem 37 (Augmentation vector) to compute a $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ such that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$. Use the algorithm supporting Theorem 43 (Sparse kernel basis) to compute a left kernel $\boldsymbol{C}$ of $\boldsymbol{v}$. Return $\boldsymbol{C}$. Correctness follows from Corollary 19.

By Theorem 37, $\boldsymbol{v}$ satisfies $||\boldsymbol{v}|| \leq n^2||\boldsymbol{A}||$. The claimed bound for $||\boldsymbol{C}||$ and the bound on the number of nonzero entries in $\boldsymbol{C}$ follow from Theorem 43. The runtime bound also follows from Theorems 37 and 43. □

To the best of our knowledge, we are not aware that the existence of such a $\boldsymbol{C}$ with only $O(n \log n)$ nonzero entries was previously known. The sparsity of $\boldsymbol{C}$ can be useful in some situations. Suppose, in addition to the full column rank input matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, we have additional linearly dependent columns $\bar{\boldsymbol{A}} \in \mathbb{Z}^{n\times m}$. If $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ is such that $\boldsymbol{CA}$ is a basis for $\boldsymbol{A}$, then a basis for the rank $n-1$ augmented matrix $\left[\begin{array}{c|c} \boldsymbol{A} & \bar{\boldsymbol{A}} \end{array}\right] \in \mathbb{Z}^{n\times((n-1)+m)}$ is given by $\boldsymbol{C}\left[\begin{array}{c|c} \boldsymbol{A} & \bar{\boldsymbol{A}} \end{array}\right]$. The additional cost to compute $\boldsymbol{C}\bar{\boldsymbol{A}}$ is only $O(nm\,\mathsf{M}(d)\log n)$ bit operations, where M is a cost function for integer multiplication, and $d = \log n + \log ||\bar{\boldsymbol{A}}||$.

6.2 Hermite lattice generator

In this section, we present a deterministic algorithm to compute a lattice basis for a matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$ via Hermite normal form method.

Theorem 54 (Hermite lattice generator)**.** *There exists an algorithm that takes as input a full column rank* $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$*, and returns as output a* $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ *such that* $\boldsymbol{C}\boldsymbol{A}$ *is a basis for* $\boldsymbol{A}$*. The output will satisfy* $||\boldsymbol{C}|| \leq n^2||\boldsymbol{A}||$*. The cost of the algorithm is* $O(n^{\omega}\,\mathsf{B}(d)(\log n)^2)$ *bit operations, where* $d = \log n + \log ||\boldsymbol{A}||$*.*

Proof. Use the algorithm supporting Theorem 37 (Augmentation vector) to compute a $\boldsymbol{v} \in \mathbb{Z}^{n\times 1}$ such that $\mathcal{L}(\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{v} \end{array}\right])$ contains the last row of $\boldsymbol{I}_n$. Use the algorithm supporting Theorem 52 (Kernel basis via Hermite) to compute a left kernel $\boldsymbol{C}$ of $\boldsymbol{v}$. Return $\boldsymbol{C}$. Correctness follows from Corollary 19.

By Theorem 37, $\boldsymbol{v}$ satisfies $||\boldsymbol{v}|| \leq n^2||\boldsymbol{A}||$. The claimed bound for $||\boldsymbol{C}||$ follows from Theorem 52. The runtime bound also follows from Theorems 37 and 52. ☐

Chapter 7

Extensions for the case: $(r+k) \times r$

Given a full column rank $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, we have given two deterministic algorithms that produces a $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ such that $\boldsymbol{C}\boldsymbol{A}$ is a basis for $\boldsymbol{A}$. Note that any such $\boldsymbol{C}$ must have at least $n-1$ nonzero entries, otherwise it would be rank deficient. We show that the $\boldsymbol{C}$ produced by our algorithm has various properties in terms of each method.

A natural direction for further research is to extend the approach to a matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-k)}$ of full column rank $n-k$. In this chapter, we show two methods to extend the two deterministic approaches to the general case.

Using the idea in the first part of the proof of Theorem 32 (Kernel of nullity one), we may firstly assume, up to permuting the rows of $\boldsymbol{A}$, that we can write $\boldsymbol{A}$ as

$$\boldsymbol{A} = \begin{bmatrix} \bar{\boldsymbol{A}} \\ \hline \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_k \end{bmatrix} \in \mathbb{Z}^{n\times(n-k)} \tag{7.1}$$

where $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(n-k)\times(n-k)}$ is nonsingular.

Section 7.1 presents an approach to compute a lattice basis generator $\boldsymbol{C} \in \mathbb{Z}^{(n-k)\times n}$ for a given full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-k)}$ by iteratively applying one of our lattice generator algorithms between Theorem 53 (Sparse Lattice generator) and 54 (Hermite lattice generator).

Section 7.2 shows how to apply Theorem 37 (Augmentation vector) iteratively and then use Theorem 29 (Kernel of matrices) to obtain the general lattice basis generator.

7.1 Method 1: Kernel of a vector

Then k applications of the algorithm supporting Theorem 53 (Sparse Lattice generator) can be used to produce a $\boldsymbol{C} \in \mathbb{Z}^{(n-k)\times n}$ such that $\boldsymbol{CA} \in \mathbb{Z}^{(n-k)\times(n-k)}$ is a basis for $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-k)}$ from (7.1).

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

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

and use Theorem 53 to find a $\bar{\boldsymbol{C}}_i$ such that $\boldsymbol{S}_i := \bar{\boldsymbol{C}}_i \bar{\boldsymbol{S}}_i$ is a basis for $\bar{\boldsymbol{S}}_i$. To combine the $\bar{\boldsymbol{C}}_i$, set

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

for $1 \le i \le k$. The final lattice basis generator is then given by

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

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

7.2 Method 2: Kernel of a matrix

From (7.1), we initialize

$$\boldsymbol{S}_1 = \left[\begin{array}{c} \bar{\boldsymbol{A}} \\ \hline \boldsymbol{a}_1 \end{array}\right] \in \mathbb{Z}^{(r+1)\times r}.$$

Then we compute $\boldsymbol{b}_1 \in \mathbb{Z}^{(r+1)\times 1}$ such that the Hermite normal form of $\left[\begin{array}{c|c} \boldsymbol{S}_1 & \boldsymbol{b}_1 \end{array}\right]$ has the last corner entry as 1 by using `DetReduction` on the matrix $\boldsymbol{S}_1 \in \mathbb{Z}^{(r+1)\times r}$.

Next, we let

$$\boldsymbol{S}_2 = \left[\begin{array}{c|c} \boldsymbol{S}_1 & \boldsymbol{b}_1 \\ \hline \boldsymbol{a}_2 & \end{array}\right] \in \mathbb{Z}^{(r+2)\times(r+1)}$$

where $\boldsymbol{S}_1$ is defined and $\boldsymbol{b}_1$ is obtained in the previous step.

Again, we compute $\boldsymbol{b}_2 \in \mathbb{Z}^{(r+2)\times 1}$ such that the Hermite norm form of $\left[\begin{array}{c|c} \boldsymbol{S}_2 & \boldsymbol{b}_2 \end{array}\right]$ has the last corner entry as 1 by using `DetReduction` on the matrix $\boldsymbol{S}_2 \in \mathbb{Z}^{(r+2)\times(r+1)}$.

After iterating the aforementioned process k times, we obtain $\boldsymbol{b}_i \in \mathbb{Z}^{(r+i)\times 1}$ for $i = 1..k$ such that the Hermite norm form of $\left[\begin{array}{c|c} \boldsymbol{S}_i & \boldsymbol{b}_i \end{array}\right]$ has the last corner entry as 1 where

$$\boldsymbol{S}_i = \left[\begin{array}{c|c} \boldsymbol{S}_{i-1} & \boldsymbol{b}_{i-1} \\ \hline \boldsymbol{a}_i & \end{array}\right] \in \mathbb{Z}^{(r+i)\times(r+i-1)}$$

We can construct a non-zeros matrix $\boldsymbol{B} \in \mathbb{Z}^{(r+k)\times k}$ where

$$\boldsymbol{B} = \left[\begin{array}{cccc} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \dots & \boldsymbol{b}_k \end{array}\right] = \begin{bmatrix} * & * & \dots & * \\ \vdots & * & \dots & * \\ * & * & \dots & * \\ & * & \dots & * \\ & & \ddots & * \\ & & & * \end{bmatrix}$$

We compute a kernel $\boldsymbol{C} \in \mathbb{Z}^{r\times(r+k)}$ of $\boldsymbol{B}^{(r+k)\times k}$ by Theorem 29 (Kernel of matrices). The algorithm then returns $\boldsymbol{C}$ as the solution to this problem.

Chapter 8

Conclusions

In this thesis, we give the simplest but not trivial extension of the scalar extended gcd problem on two integers to the case of integer input matrices. We present a deterministic algorithm that takes as input a full column rank $\boldsymbol{A} \in \mathbb{Z}^{n\times(n-1)}$, and returns as output a matrix $\boldsymbol{C} \in \mathbb{Z}^{(n-1)\times n}$ such that $\boldsymbol{B} := \boldsymbol{C}\boldsymbol{A} \in \mathbb{Z}^{(n-1)\times(n-1)}$ satisfies $\mathcal{L}(\boldsymbol{B}) = \mathcal{L}(\boldsymbol{A})$. The matrix $\boldsymbol{C}$ is sparse, with at most $O(n \log n)$ nonzero entries, and satisfies $\log ||\boldsymbol{C}|| \in \Theta(\log n + \log ||\boldsymbol{A}||)$. More precisely, we show that at most $n(1 + \log_2 n)$ entries of $\boldsymbol{C}$ will be nonzero, and that $\log ||\boldsymbol{C}|| \leq 4 \log n + 2 \log ||\boldsymbol{A}||$. The cost of the algorithm is $O(n^\omega \mathsf{B}(d)(\log n)^2)$ bit operations, where $d = \log n + \log ||\boldsymbol{A}||$ and ω is the exponent of matrix multiplication. This cost estimate is about the same as that required to multiply together two square integer matrices of dimension n that have magnitude of entries bounded by $||\boldsymbol{A}||$.

Our algorithm to compute the basis $\boldsymbol{B}$ improves by about a factor of n on the fastest previous algorithm presented by Li and Nguyen [2019, Theorem 3.8] which requires $O(n^\omega \mathsf{B}(nd))$ bit operations. We remark that our algorithm can also be used to handle the general case in a Las Vegas fashion: given a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, find a matrix $\boldsymbol{B} \in \mathbb{Z}^{m\times m}$ such that $\mathcal{L}(\boldsymbol{A}) = \mathcal{L}(\boldsymbol{B})$. The steps are as follows:

1. First, we use the Monte Carlo technique in Chen and Storjohann [2005] to compress from $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ to $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(m+1)\times m}$.

2. Next, we use our algorithm to compress from $\bar{\boldsymbol{A}} \in \mathbb{Z}^{(m+1)\times m}$ to $\boldsymbol{B} \in \mathbb{Z}^{m\times m}$.

3. Finally, we can use integrality certification by Birmpilis et al. [2023, Theorem 17 (3.4)] to assay if $\boldsymbol{A}\boldsymbol{B}^{-1}$ is integral. If $\boldsymbol{A}\boldsymbol{B}^{-1}$ is integral, then $\boldsymbol{B} \in \mathbb{Z}^{m\times m}$ is a basis for $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$. Otherwise, we repeat step 1.

The Las Vegas algorithm just described has expected running time $O(nm^{\omega-1}\log||\boldsymbol{A}||)$ bit operations and produces a basis $\boldsymbol{B}$ with $\log||\boldsymbol{B}|| \in O(\log n + \log||\boldsymbol{A}||)$.

We remark that the algorithm of Li and Nguyen [2019] also handles the more general case of the problem: given an $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, it can compute a basis $\boldsymbol{B} \in \mathbb{Z}^{m\times m}$ for $\boldsymbol{A}$ in $O\tilde{}(nm^{\omega}\log||\boldsymbol{A}||)$ bit operations. A natural direction for further research is whether this cost estimate can be improved by a factor of m. We end with the following open problem.

Open Problem 1. *Given a full column rank matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$, compute a basis $\boldsymbol{B} \in \mathbb{Z}^{m\times m}$ for $\boldsymbol{A}$ that satisfies $\log||\boldsymbol{B}|| \in O(n+\log||\boldsymbol{A}||)$ in $O\tilde{}(nm^{\omega-1}\log||\boldsymbol{A}||)$ bit operations.*

References

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

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.

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.

Z. Chen and A. Storjohann. Lattice compression of integer matrices. *Journal of Symbolic Computation*, 2005.

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

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.

D. Harvey and J. van der Hoeven. Integer multiplication in time $O(n \log n)$. *Annals of Mathematics*, 193(2):563–617, 2021.

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.

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.

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

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

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

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

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.