Hermite Forms of Polynomial Matrices

by

Somit Gupta

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, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents a new algorithm for computing the Hermite form of a polynomial matrix. Given a nonsingular $n \times n$ matrix A filled with degree d polynomials with coefficients from a field, the algorithm computes the Hermite form of A using an expected number of $(n^\omega d)^{1+o(1)}$ field operations. The algorithm is randomized of the Las Vegas type.

Acknowledgements

I am greatly indebted to my supervisor professor Arne Storjohann for introducing me to the world of symbolic computation and providing the direction for my research. I have benefited a lot from his attention to detail and planning, and emphasis on clarity of presentation. I hope I will be carrying back some these qualities with me.

I am grateful to my readers professor George Labahn and professor Mark Giesbrecht for taking time out to read the thesis and provide their valuable feedback. Their friendly advise on research, presentation and writing has been very helpful to me.

I want to express my gratitude to all my teachers who have all always encouraged my inquisitiveness and helped me take it further. I thank professor R. B. Bapat for helping me take my first steps in research and introducing me to the area of matrix algebra.

Finally, I thank my family and friends for their constant support and encouragement, without which it would not have been possible to complete this thesis.

Dedication

This thesis is dedicated to my mother *Vinod Bala Gupta.*

Contents

List of Figures

Chapter 1

Introduction

Among the classical normal forms for matrices over a principal ideal domain, the Hermite form is the best known. Recall the definition of the form over the ring $\mathsf{K}[x]$ of univariate polynomials over a field K. Corresponding to any nonsingular $A \in \mathsf{K}[x]^{n\times n}$ is a unimodular matrix $U \in \mathsf{K}[x]^{n\times n}$ such that

$$H = UA = \begin{bmatrix} h_1 & \bar{h}_{12} & \cdots & \bar{h}_{1n} \\ & h_2 & \cdots & \bar{h}_{2n} \\ & & \ddots & \vdots \\ & & & h_n \end{bmatrix}$$

is upper triangular, h_j is monic for $1 \leq j \leq n$, and $\deg \bar{h}_{ij} < \deg h_j$ for $1 \leq i < j \leq n$. The problem of computing the Hermite form has received a considerable amount of attention. For example, the theses [5, 25, 22, 35, 6, 26] and ISSAC papers [33, 31, 34, 23] have addressed this topic.

Modulo determinant algorithms [14, 16, 8], see also [4], compute the Hermite form of A working modulo the determinant and require $O^{\sim}(n^4 d)$ field operations from K. Matrix multiplication can be introduced [14, 31] to reduce the cost to $O^{\sim}(n^{\omega+1}d)$, where $2 < \omega \leq 3$ is the exponent of matrix multiplication. The iterative approach in [24] gives a deterministic $O(n^3 d^2)$ algorithm, achieving a running time that is cubic in n but at the cost of increasing the exponent of d to two. In [13] we give a Las Vegas algorithm to compute H using an expected number of $O^{\sim}(n^3 d)$ field operations from K. In this thesis we give a Las Vegas algorithm to compute H using an expected number of $O^{\sim}(n^{\omega} d)$ field operations from K.

Our result closes the time complexity gap between the computation of the Hermite form and many problems on polynomial matrices that have an algorithm which costs $O^{\sim}(n^{\omega} d)$ field operations. Examples include the high-order lifting based linear solving, determinant and Smith form algorithms in [27, 28], the fast row reduction algorithm of [11] and minimal

approximant basis algorithms in [11, 36]. The techniques in [21] can be adapted to the case of polynomial matrices and achieve algorithms that are subcubic in n for many problems. It is even known that the explicit inverse of A, which has total size $\Omega(n^3 d)$, can be computed in nearly optimal time $O^{\sim}(n^3 d)$ [18, 30].

The rest of the chapter is organised in the following manner. Subsection 1.1 defines the cost model over which we work and the cost of some common operations. In Subsection 1.2, we introduce some preliminaries and recall a few definitions and results which will be used often in the thesis. Subsection 1.3 gives the overview of the thesis.

1.1 Cost Model and Operations over $\mathsf{K}[x]$

Algorithms are analyzed by bounding the number of required field operations from a field K on an algebraic random access machine; the operations $+$, $-$, $\times$ and "divide by a nonzero" involving two field elements have unit cost.

We use the cost function M for polynomial multiplication: let $\mathsf{M}\colon \mathbb{Z}_{\geq 0} \to \mathbb{R}_{>0}$ be such that polynomials in $\mathsf{K}[x]$ of degree bounded by d can be multiplied using at most $\mathsf{M}(d)$ field operations from K. The operation of multiplying by a power of x is free. We shall often use the fact that M is a super-linear function:

$$\mathsf{M}(ab) \leq \mathsf{M}(a)\mathsf{M}(b), \tag{1.1}$$

for $a, b \in \mathbb{Z}_{>0}$. Note that superlinearity also implies that for any $t \in \mathbb{Z}_{>0}, t \in O(\mathsf{M}(t))$.

We use the cost function MM for matrix multiplication: let $\mathsf{MM}\colon \mathbb{Z}_{>0} \to \mathbb{R}_{>0}$ be such that two matrices, of dimensions bounded by n, filled with entries from K can be multiplied using at most $\mathsf{MM}(n)$ field operations from K. Throughout the thesis we shall assume that

$$\mathsf{MM}(n) \in O(n^{\omega}), \tag{1.2}$$

where $2 < \omega \leq 3$ denotes the exponent of matrix multiplication. As can be seen from (1.2) MM is super-linear:

$$\mathsf{MM}(ab) \leq \mathsf{MM}(a)\mathsf{MM}(b), \tag{1.3}$$

for $a, b \in \mathbb{Z}_{>0}$. In fact we assume that MM is super-quadratic function:

$$\mathsf{MM}(n) \in \Omega(n^{2+\epsilon}),$$

for some $\epsilon > 0$.

For multiplication of rectangular matrices, let $\mathsf{MM}\colon \mathbb{Z}_{>0} \times \mathbb{Z}_{>0} \times \mathbb{Z}_{>0} \to \mathbb{R}_{>0}$ be such that a scalar matrix $\in \mathsf{K}^{a \times b}$ can be multiplied with a scalar matrix $\in \mathsf{K}^{b \times c}$ in cost bounded by $\mathsf{MM}(a, b, c)$. Using an obvious block decomposition we get

$$\mathsf{MM}(a, b, c) \leq \lceil a/r \rceil \cdot \lceil b/r \rceil \cdot \lceil c/r \rceil \cdot \mathsf{MM}(r), \tag{1.4}$$

where $r = \min(a, b, c)$.

Now consider the multiplication of polynomial matrices. The degree of a matrix over $\mathsf{K}[x]$ is the maximum of the degrees of the entries. Let $\mathsf{MM}: \mathbb{Z}_{>0} \times \mathbb{Z}_{\geq 0} \to \mathbb{R}_{>0}$ be such that two matrices over $\mathsf{K}[x]$, of dimensions bounded by n and degree at most d, can be multiplied using at most $\mathsf{MM}(n, d)$ field operations from K. We can similarly define $\mathsf{MM}(a, b, c, d)$ as the bound for the cost of multiplication of a matrix $\in \mathsf{K}[x]^{a \times b}$ by a matrix $\in \mathsf{K}[x]^{b \times c}$, where the degrees of the matrices are bounded by d. We shall often use the fact we can multiply two polynomial matrices using the procedure to multiply two scalar matrices but replacing the scalar operations with polynomial operations. This gives

$$\mathsf{MM}(n, d) \leq \mathsf{MM}(n)\,\mathsf{M}(d) \quad \text{and} \quad \mathsf{MM}(a, b, c, d) \leq \mathsf{MM}(a, b, c)\,\mathsf{M}(d). \tag{1.5}$$

We refer to [10] for more details and references about ω and M. Some of our complexity estimates will explicitly make the assumption that

$$\mathsf{M}(t) \in O(t^{\omega-1}). \tag{1.6}$$

This assumption states that if fast matrix multiplication techniques are used, then fast polynomial multiplication should also be used.

Given two polynomials $a, b \in \mathsf{K}[x]$ with b nonzero, we use the subroutines $\mathrm{Rem}(a, b)$ and $\mathrm{Quo}(a, b)$ in our algorithms to return the unique polynomials such that $a = \mathrm{Quo}(a, b)\, b + \mathrm{Rem}(a, b)$ with $\deg \mathrm{Rem}(a, b) < \deg b$. If a and b have degree bounded by d then the Rem operation has cost $O(\mathsf{M}(d))$ and Quo operation has cost $O(\mathsf{M}(\deg a - \deg b)) = O(\mathsf{M}(\deg \mathrm{Quo}(a, b)))$. Note that if b is a power of x then both the Rem and Quo operation are free in our cost model. If the first argument of Rem or Quo is a matrix or vector the intention is to apply the operation element wise to the entries.

The gcd problem takes as input two polynomials $a, b \in \mathsf{K}[x]$ and asks as output the monic greatest common divisor of a and b. The extended gcd problem takes as input two polynomials $a, b \in \mathsf{K}[x]$, and asks as output polynomials $g, s, t, u, v \in \mathsf{K}[x]$ such that

$$\begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix}, \tag{1.7}$$

with g as the monic greatest common divisor of a and b, and $sv - tu$ a nonzero constant polynomial. In our algorithms, we denote by $\mathrm{Gcd}(a, b)$ and $\mathrm{Gcdex}(a, b)$ the subroutines that return g and the tuple (g, s, t, u, v) as described above.

It will be useful to define an additional cost function B to bound the cost of the extended gcd operation, as well as other gcd-related computations. We can take

$$\mathsf{B}(d) \leq \min\{\mathsf{M}(d) \log d,\ d^2\}. \tag{1.8}$$

Then the extended gcd problem with two polynomials in $\mathsf{K}[x]$ of degree bounded by d can be solved in time $O(\mathsf{B}(d))$. Due to the superlinearity of M, (1.1), and (1.8) B is super-linear:

$$\mathsf{B}(ab) \leq \mathsf{B}(a)\mathsf{B}(b) \tag{1.9}$$

for $a, b \in \mathbb{Z}_{>0}$.

1.1.1 Operations over $\mathsf{K}[x]/\langle s \rangle$

Throughout the thesis we shall deal with two kinds of algebraic structures: the principal ideal domain (PID) $\mathsf{K}[x]$, and principal ideal rings (PIR) of the form $\mathsf{R} = \mathsf{K}[x]/\langle s \rangle$, $s \in \mathsf{K}[x]$ nonzero. A PIR is a nonzero commutative ring whose ideals are principal, i.e., can be generated by a single element. A PID is a PIR with no zero divisors. In the previous subsection we have already defined the operations Rem, Quo, Gcd and Gcdex over $\mathsf{K}[x]$. In this subsection we define some additions operations and show how their analogues can be computed over the residue class ring $\mathsf{K}[x]/\langle s \rangle$.

Let R be a principal ideal ring (either $\mathsf{K}[x]$ or $\mathsf{K}[x]/\langle s \rangle$). An element $a \in \mathsf{R}$ is said to be a unit if and only if there exists an element $b \in \mathsf{R}$ such that $ab = ba = 1$. In other words $a \in \mathsf{R}$ is a unit if and only if it is invertible. Two elements $a_1, a_2 \in \mathsf{R}$ are said to be associates of each other if and only if there exists a unit $u \in \mathsf{R}$ such that $a_1 = ua_2$. A set of elements of R, one from each associate class, is called a prescribed complete set of non-associates; we denote such a set by $\mathcal{A}(\mathsf{R})$. Two elements $a_1, a_2 \in \mathsf{R}$ are said to be congruent to each other with respect to another element $0 \neq b \in \mathsf{R}$ if and only if $b \mid (a_1 - a_2)$. Corresponding to an element $b \in \mathsf{R}$, a set of elements, one from each congruence class, is said to be a prescribed complete set of residues with respect to b; we denote such a set by $\mathcal{R}(\mathsf{R}, b)$.

The natural and canonical choice for $\mathcal{A}$ and $\mathcal{R}$ over $\mathsf{K}[x]$ are $\mathcal{A}(\mathsf{K}[z]) = \{0\} \cup \{f \in \mathsf{K}[z] \mid f \text{ is monic}\}$ and $\mathcal{R}(\mathsf{K}[z], s) = \{f \in \mathsf{K}[z] \mid \deg f < \deg s\}$. For a nonzero s, we can identify the residue class ring $\mathsf{K}[x]/\langle s \rangle$ with the set of elements $\mathcal{R}(\mathsf{K}[x], s)$, and define

$$\mathcal{A}(\mathsf{K}[x]/\langle s \rangle) = \{\gcd(a, s) \,|\, a \in \mathrm{Rem}(\mathsf{K}[x], s)\}$$

and

$$\mathcal{R}(\mathsf{K}[x]/\langle s \rangle, b) = \mathcal{R}(\mathsf{K}[x], \gcd(b, s)).$$

For a $a \in \mathsf{R}$, the operations $\mathrm{Ass}(a)$ returns the associate of a in the proscribed set of associates $\mathcal{A}(\mathsf{R})$, and the operation $\mathrm{Unit}(a)$ returns a unit $u \in \mathsf{R}$ such that $\mathrm{Ass}(a) = ua \in \mathcal{A}(\mathsf{R})$. For $\mathsf{R} = \mathsf{K}[x]$ and $a \in \mathsf{K}[x]$, $\mathrm{Unit}(a)$ returns 1 if a is zero, and the inverse of the leading coefficient of a otherwise.

We shall often pass back and forth between $\mathsf{K}[x]$ and its residue class ring $\mathsf{K}[x]/\langle s \rangle$. A homomorphism $\phi_s : \mathsf{K}[x] \longrightarrow \mathsf{K}[x]/\langle s \rangle$ can be naturally defined as $\phi_s(a) = \mathrm{Rem}(a, s)$ for

$a \in \mathsf{K}[x]$. We choose to denote by $\phi_s^{-1} : \mathsf{K}[x]/\langle s\rangle \longrightarrow \mathsf{K}[x]$, such that for any $\bar{a} \in \mathsf{K}[x]$, $\phi_s^{-1}(\bar{a})$ is the unique element of $\mathcal{R}(\mathsf{K}[x], s)$ that maps to b under ϕ. For example if $\bar{a} = 3x + 1 \in \mathsf{K}[x]/\langle x^3 + x\rangle$, then $\phi^{-1}_{x^3+x}(3x + 1) = 3x + 1 \in \mathsf{K}[x]$.

For $\bar{a}, \bar{b} \in \mathsf{K}[x]/\langle s\rangle$, let $a = \phi_s^{-1}(\bar{a})$ and $b = \phi_s^{-1}(\bar{b})$. We can do the following basic operations over $\mathsf{K}[x]/\langle s\rangle$ by translating them into operations over $\mathsf{K}[x]$:

- $\bar{a} \diamond \bar{b} = \phi_s(\mathrm{Rem}(a \diamond b, s))$, where $\diamond \in \{+, -, *\}$.
- $\bar{a}/\bar{b} = \phi_s(a/b)$. This operation is valid only if $b \mid a$.
- $\mathrm{Unit}(\bar{a}) = \left[\begin{array}{l} (*, u, *, *, *) = \mathrm{Gcdex}(a, s) \\ \textbf{return } \phi_s(u). \end{array}\right.$
- $\mathrm{Ass}(\bar{a}) = \mathrm{Unit}(\bar{a}) * \bar{a}$.
- $\mathrm{Gcd}(\bar{a}, \bar{b}) = \mathrm{Ass}(\phi_s(\mathrm{Gcd}(a, b)))$.
- $\mathrm{Rem}(\bar{a}, \bar{b}) = \phi_s(\mathrm{Rem}(a, \mathrm{Gcd}(b, s)))$.
- $\mathrm{Quo}(\bar{a}, \bar{b}) = \mathrm{Unit}(\bar{b})((\bar{a} - \mathrm{Rem}(\bar{a}, \bar{b}))/\mathrm{Ass}(\bar{b}))$.

Thus we see that that any basic operation in $\mathsf{K}[x]/\langle s\rangle$ has a cost bounded by $O(\mathsf{B}(\deg s))$ basic operations from K.

1.2 Preliminaries

We shall be working primarily with square matrices. For any square matrix A of dimension n, we denote by $A[i, j]$ the i, j entry of A, $1 \le i, j \le n$. We will often use $*$ to denote an index which takes values from the entire domain. For example $A[*, i]$ implies the i-th column of A. We denote by $A_{[\mathcal{S}|\mathcal{T}]}$ the submatrix of A comprised of the intersection of only those rows of A whose index is in the list $\mathcal{S}$ and only those columns of A whose index is in the list $\mathcal{T}$. For $A \in \mathsf{K}[x]^{n\times n}$, we denote $\Delta_i(A)$ to be the gcd of all $i \times i$ minors of A and $\Delta_i^c(A)$ to be the gcd of all $i \times i$ minors of the first i columns of A. Let $\Delta_0(A) = \Delta_0^c(A) = 1$.

Next we describe two normal forms of matrices over $\mathsf{K}[x]$ and some of their properties that we shall use often.

1.2.1 Hermite Form

Definition 1.1. *A matrix $H \in \mathsf{K}[x]^{n\times n}$ is said to be in Hermite form if H can be written as*

$$\begin{bmatrix} h_1 & \bar{h}_{12} & \bar{h}_{13} & \cdots & \bar{h}_{1n} \\ & h_2 & \bar{h}_{23} & \cdots & \bar{h}_{2n} \\ & & h_3 & \cdots & \bar{h}_{3n} \\ & & & \ddots & \vdots \\ & & & & h_n \end{bmatrix} \in \mathsf{R}^{n\times n},$$

with

- *h_i is monic for all $1 \le i \le n$, and*
- *$\deg \bar{h}_{ij} < \deg h_j$ for $1 \le i < j \le n$.*

Corresponding to every nonsingular $A \in \mathsf{K}[x]^{n\times n}$, there exists a unimodular $U \in \mathsf{K}[x]^{n\times n}$ and the unique $H \in \mathsf{K}[x]^{n\times n}$ in Hermite form such that

$$H = UA.$$

H is called the *Hermite Form* of A; $\mathrm{hnf}(A) = H$.

Theorem 1.2. *[19] For the Hermite form $H \in \mathsf{K}[x]^{n\times n}$ of a nonsingular $A \in \mathsf{K}[x]^{n\times n}$, the following properties hold:*

- *$h_i^* = \prod_{j=1}^{i} h_j = \Delta_i^c(A)$, for all $1 \le i \le n$,*
- *$h_i = \Delta_i^c(A)/\Delta_{i-1}^c(A)$, for all $1 \le i \le n$, and*
- *$\deg h_i \le i \deg A$, for all $1 \le i \le n$,*

where $\Delta_i^c(A)$ is the gcd *of all $i \times i$ minors of the first i columns of A.*

1.2.2 Smith Form

Definition 1.3. *A matrix $S \in \mathsf{K}[x]^{n\times n}$ is said to be in Smith form if S can be written as $\mathrm{diag}(s_1, s_2, s_3, \ldots, s_n) \in \mathsf{R}^{n\times n}$ with*

- *s_i is monic for all $1 \le i \le n$, and*
- *$s_i \mid s_{i+1}$ for all $1 \le i < n$.*

Corresponding to every $A \in \mathsf{R}^{n\times n}$ there exist unimodular matrices $P, Q \in \mathsf{R}^{n\times n}$, and the unique $S \in \mathsf{R}^{n\times n}$ in Smith form such that

$$S = PAQ.$$

S is called the *Smith Form* of A; $\mathrm{snf}(A) = S$.

Theorem 1.4. *[19] For the Smith form $S \in \mathsf{K}[x]^{n\times n}$ of a nonsingular $A \in \mathsf{K}[x]^{n\times n}$, the following properties hold:*

- $s_i^* = \prod_{j=1}^{i} s_j = \Delta_i(A)$, *for all* $1 \le i \le n$,
- $s_i = \Delta_i(A)/\Delta_{i-1}(A)$, *for all* $1 \le i \le n$, *and*
- $\deg s_i \le \frac{n}{n-i+1} \deg A$, *for all* $1 \le i \le n$,

where $\Delta_i(A)$ is the gcd *of all $i \times i$ minors of A.*

Note that Smith form is also a canonical form over $\mathsf{R} = \mathsf{K}[x]/\langle s\rangle$ and for the Smith form $S \in R^{n\times n}$ of a $A \in R^{n\times n}$, s_1 divides all the entries of A.

1.3 Overview

We present a brief overview of our algorithm developed in the thesis. Given a nonsingular $A \in \mathsf{K}[x]^{n\times n}$ with the degree of each element bounded by d, we aim to develop a Las Vegas algorithm to compute the Hermite form H of A in cost $O^{\sim}(n^{\omega}d)$.

The main difficulty to obtain fast algorithms for the Hermite form seems to be the unpredictability and non uniformity of the degrees of the diagonal entries. The best *a priori* bound for $\deg h_j$ is jd, $1 \le j \le n$. Summing these a priori bounds gives $\sum_{j=1}^{n} jd \in \Theta(n^2 d)$, which overshoots by a factor of n the *a priori* bound $\sum_{j=1}^{n} \deg h_j = \det A \le nd$. For comparison, for the diagonal Smith form $S := \bar{U}A\bar{V} = \mathrm{Diag}(s_1, \ldots, s_n)$ of A, a canonical form under left and right unimodular multiplication, we have the *a priori* bounds $\deg s_j \le (n/(n-j+1))d$; summing these yields $\Theta(nd \log n)$. These good *a priori* bounds for the invariant factors s_j are exploited, for example, in [27, 9] to get improved algorithms for computing the Smith form.

Consider the following nonsingular matrix

$$A = \begin{bmatrix} [d] & [d] & [d] & [d] \\ [d] & [d] & [d] & [d] \\ [d] & [d] & [d] & [d] \\ [d] & [d] & [d] & [d] \end{bmatrix} \in \mathsf{K}[x]^{4\times 4},$$

where $[d]$ represents a polynomial with degree bounded by d. The *a priori* bounds for the entries of the Hermite form of A are the following:

$$\begin{bmatrix} [d] & (2d) & (3d) & (4d) \\ & [2d] & (3d) & (4d) \\ & & [3d] & (4d) \\ & & & [4d] \end{bmatrix},$$

where (t) represents a polynomial of degree strictly less than t. The *a priori* bounds for the entries of the Smith form of A are the following:

$$\begin{bmatrix} [d] & & & \\ & [4/3\ d] & & \\ & & [2d] & \\ & & & [4d] \end{bmatrix}.$$

Thus, while the degree bounds for the diagonal entries of the Hermite form decrease in a arithmetic progression from the last diagonal entry to the first, the degree bounds for the diagonal entries of the Smith form decrease faster in a harmonic progression.

The key to our approach in the thesis is to use the Smith form S of A, together with partial information of a right unimodular transform $\bar{V}$, in order to obtain the Hermite form H of A. Our algorithm has four main phases.

The first phase computes a row reduced form of the input matrix A. We use existing algorithms [12] to compute the row reduced form.

The second phase is to compute the Smith form S, and V, such that $A \equiv USV \mod s_n$. In Chapter 3, we modify the `OuterProductAdjoint` algorithm of [30] by working over blocks of columns of A instead of one column at a time. The major hurdle in reducing the cost of this phase to $O\tilde{}(n^\omega\, d)$ is carrying out an operation of the type $\mathrm{Rem}(\mathrm{Rem}(FG, s), x^d)$ where $F \in \mathsf{K}[x]^{n\times 1}, G \in \mathsf{K}[x]^{1\times n}$ with degree of each term in F and G bounded by $\deg s \leq nd$.

For example, consider the following matrix with the Smith form $\mathrm{diag}(s_1, s_2, \cdots, s_n)$:

$$A = \begin{bmatrix} [d] & [d] & \cdots & [d] \\ [d] & [d] & \cdots & [d] \\ \vdots & \vdots & \vdots & \vdots \\ [d] & [d] & [d] & [d] \end{bmatrix} \in \mathsf{K}[x]^{n\times n}.$$

The outer product decomposition of this matrix will look like the following:

$$s_n A^{-1} \equiv \begin{bmatrix} [nd] \\ [nd] \\ \vdots \\ [nd] \end{bmatrix} \begin{bmatrix} [nd] & [nd] & \cdots & [nd] \end{bmatrix} + s_n/s_{n-1} * \mod s_n.$$

In the course of computing a reduced Smith decomposition of A, we need to compute entities of the following shape and bounds on the degree of the entries:

$$\mathrm{Rem}\left(\mathrm{Rem}\left(\overbrace{\begin{bmatrix}[nd]\\ [nd]\\ \vdots\\ [nd]\end{bmatrix}}^{F}\overbrace{\begin{bmatrix}[nd] & [nd] & \cdots & [nd]\end{bmatrix}}^{G}, s_n\right), x^d\right).$$

Notice that the size of the input and output in this operation is bounded by n^2d, but we are not aware of any known algorithm which can compute this in $O\tilde{}(n^\omega\, d)$ operations from K. The direct way to perform the above operation is to first compute $\mathrm{Rem}(FG, s_n)$ and then reduce it modulo x^d. The direct method costs $O(n^3d)$ basic operations from K.

In Chapter 2, we develop a procedure `PartialProduct` to compute $\mathrm{Rem}(\mathrm{Rem}(FG, s), x^d)$ in $O\tilde{}(n^\omega d)$ operations from K. The main idea is to use the Sylvester matrix representation for multiplying two polynomials. For example consider the following operation:

$$\mathrm{Rem}\left(\mathrm{Rem}\left(\overbrace{\begin{bmatrix}[2d]\\ [2d]\end{bmatrix}}^{F}\overbrace{\begin{bmatrix}[2d] & [2d]\end{bmatrix}}^{G}, [2d]\right), x^d\right).$$

We transform it to the following:

$$\mathrm{Rem}\left(\mathrm{Rem}\left(\begin{bmatrix}1 & x^d & x^{2d} & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & x^d & x^{2d}\end{bmatrix}\begin{bmatrix}(d) & \\ (d) & (d)\\ & (d)\\ (d) & \\ (d) & (d)\\ & (d)\end{bmatrix}\begin{bmatrix}(d) & (d)\\ (d) & (d)\end{bmatrix}, [2d]\right), x^d\right).$$

After some preprocessing the above procedure reduces to two matrix multiplications of matrices with degree of entries bounded by d.

The third phase is to compute the degrees of the diagonal entries of H. In Chapter 4, We show that this can be accomplished via a unimodular matrix triangularization:

$$\left[\begin{array}{c|c} S & \\ \hline V & I_n\end{array}\right] \longrightarrow \left[\begin{array}{c|c} I_n & * \\ \hline & T\end{array}\right] \in \mathsf{K}[x]^{2n\times 2n}. \tag{1.10}$$

The matrix V is obtained from $\bar{V}$ by reducing entries in column j modulo s_j, $1 \le j \le n$. We show that the submatrix T in (1.10) will be left equivalent to A and thus, up to associates,

has the same diagonal entries as H. When performing the triangularization in (1.10), we exploit the fact that S is diagonal by keeping off-diagonal entries in the first n columns of the work work matrix reduced modulo the diagonal entry in the same column. Using the upper bound $\sum_{j=1}^{n} \deg s_j \leq nd$, and by avoiding explicit computation of the off-diagonal entries of T and the block above T, we can compute the diagonal entries of T in $O^{\sim}(n^{\omega}\, d)$ operations from K.

Chapter 5 describes the fourth phase of our algorithm. It uses the knowledge of the degrees of the diagonal entries of H to set up a minimal approximant basis problem for recovering H. In particular, the Hermite form H can recovered by computing a left kernel basis in canonical form for the first n columns of the matrix in (1.10):

$$\left[\ -HVS^{-1} \mid H\ \right] \left[\frac{S}{V}\right] = 0.$$

We show how to transform the kernel computation shown above to an equivalent problem that can be solved in time $O(n^{\omega}\, \mathsf{B}(d))$ using the fast minimal approximant basis algorithm of [11]. Our problem transformation makes use of the partial linearization and reduction of order techniques in [29].

Chapter 6 sums up the whole algorithm and recalls the main result of the thesis. It also discusses some future directions of research.

Chapter 2

Partial Product

Let K be a field. Let $A \in \mathsf{K}[x]^{n\times\ell}, B \in \mathsf{K}[x]^{l\times n}$, and $s, h \in \mathsf{K}[x]$. We want to compute

$$\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d).$$

Our target cost in terms of number of basic operations from K is $O(\mathsf{MM}(n, d))$ subject to the conditions that $\deg B, \deg h < \deg s$, and $l \deg A, \ell \deg B, \deg s \in O(nd)$. If we measure the size of the input and output in terms of the number of elements from K required to describe the input, it is interesting to note that in this case even though the size of the input and the output is bounded by $O(n^2 d)$, the direct method which computes $\mathrm{Rem}(hAB, s)$ as an intermediate quantity would require $\Omega(n^3 d)$ operations to compute $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d)$ since the size of $\mathrm{Rem}(hAB, s)$ can potentially be of the order $O(n^3 d)$. Consider the following example of solving the problem by the conventional method.

Example 2.1. *Let* $\mathsf{K} = \mathbb{Z}_7$,

$$A = \begin{bmatrix} 4\,x^2 + 2\,x + 1 \\ 3\,x^2 + x + 5 \\ x^2 + 2 \end{bmatrix}, B = \begin{bmatrix} x^2 + 3 & 5\,x^2 + x + 4 & 3\,x^2 + 6 \end{bmatrix},$$

$s = x^6 + 5\,x^5 + 3\,x^4 + 2\,x + 2$, $h = 5\,x^5 + 3\,x^4 + 2\,x^3 + x^2 + 3\,x$, *and* $d = 2$.

To find $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^2)$, *one would first find the product*

$$AB = \begin{bmatrix} 4\,x^4 + 2\,x^3 + 6\,x^2 + 6\,x + 3 & 6\,x^4 + 2\,x^2 + 2\,x + 4 & 5\,x^4 + 6\,x^3 + 6\,x^2 + 5\,x + 6 \\ 3\,x^4 + x^3 + 3\,x + 1 & x^4 + x^3 + 3\,x^2 + 2\,x + 6 & 2\,x^4 + 3\,x^3 + 5\,x^2 + 6\,x + 2 \\ x^4 + 5\,x^2 + 6 & 5\,x^4 + x^3 + 2\,x + 1 & 3\,x^4 + 5\,x^2 + 5 \end{bmatrix}$$

and then multiply it by h and reduce mod s as
$\mathrm{Rem}(hAB, s) =$

$$\begin{bmatrix} 5\,x^5 + x^4 + 6\,x^3 + 3\,x^2 + 6\,x + 3 & 6\,x^5 + 3\,x^4 + 3\,x^3 + 6\,x + 2 & 4\,x^5 + 3\,x^4 + 5\,x^3 + 3\,x^2 + 4\,x + 3 \\ 6\,x^5 + 6\,x^4 + 6\,x^3 + 6\,x + 2 & 2\,x^5 + x^4 + 5\,x^3 + 6\,x^2 + 4\,x + 5 & 4\,x^5 + 4\,x^4 + 3\,x^2 + 5\,x + 4 \\ x^4 + 5\,x^3 + 2\,x^2 + 4\,x + 6 & 6\,x^5 + x^4 + 3\,x^3 + 5\,x^2 + 5\,x + 4 & x^5 + x^4 + x^3 + 2\,x^2 + 4\,x + 5 \end{bmatrix}.$$

Finally, we get

$$\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^2) = \begin{bmatrix} 6\,x + 3 & 6\,x + 2 & 4\,x + 3 \\ 6\,x + 2 & 4\,x + 5 & 5\,x + 4 \\ 4\,x + 6 & 5\,x + 4 & 4\,x + 5 \end{bmatrix}.$$

In this chapter, we shall discuss a new method to compute $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d)$ efficiently, without calculating $\mathrm{Rem}(hAB, s)$ explicitly.

To explain the different ideas used we proceed in three subsections. For $f, g, s \in \mathsf{K}[x]$ and $d \in \mathbb{Z}_{>0}$, Subsection 2.1 presents the algorithm to reduce the operation of computing $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$ to that of computing the dot product of two polynomial vectors with degree bounded by $O(d)$. In Subsection 2.2, we extend the idea of the previous subsection to accommodate vectors $F \in \mathsf{K}[x]^{n\times 1}$ and $G \in \mathsf{K}[x]^{1\times n}$, reducing the computation of $\mathrm{Rem}(\mathrm{Rem}(FG, s), x^d)$ to that of multiplying together two matrices degree $O(d)$. Finally, for $A \in \mathsf{K}[x]^{n\times \ell}, B \in \mathsf{K}[x]^{l\times m}, h \in \mathsf{K}[x]$, Subsection 2.3 reduces the operation of computing $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d)$ to polynomial matrix multiplication.

Section 2.4 gives a generalization of the technique described up to this point in the chapter. For any $k \geq 0$, not just $k = 0$, we show how to find the coefficient of X^k in the X-adic expansion of $\mathrm{Rem}(hAB, s)$, $X = x^d$.

2.1 Scalar Case: Computing $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$

Let $f, g, s \in \mathsf{K}[x]$ and $d \in \mathbb{Z}_{>0}$. In this section we show how to transform the problem of computing $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$ to that of computing two dot products of vectors with degree bounded by $O(d)$.

We can write $f, g \in \mathsf{K}[x]$ uniquely in x^d-adic expansion as

$$f = f_0 + f_1 x^d + f_2 x^{2d} + \cdots + f_t x^{td} \tag{2.1}$$

and

$$g = g_0 + g_1 x^d + g_2 x^{2d} + \cdots + g_u x^{ud}, \tag{2.2}$$

where $u = \lfloor (\deg f)/d \rfloor$, $t = \lfloor (\deg g)/d \rfloor$, and $\deg f_i < d$, $\deg g_j < d$ for $0 \leq i \leq t$ and $0 \leq j \leq u$. Let $X = x^d$,

$$\hat{H} = [\ 1 \quad X \quad X^2 \quad \cdots \quad X^{u+t-1} \quad X^{u+t}\ ], \tag{2.3}$$

$$G = [\ g_0 \quad g_1 \quad g_2 \quad \cdots \quad g_{u-1} \quad g_u\ ]^T, \tag{2.4}$$

and

$$F = \begin{bmatrix} f_0 & & & \\ f_1 & f_0 & & \\ & f_1 & \ddots & \\ \vdots & \vdots & & f_0 \\ & & & f_1 \\ f_t & & & \\ & f_t & & \vdots \\ & & \ddots & \\ & & & f_t \end{bmatrix} \in \mathsf{K}[x]^{(u+t)\times u}. \tag{2.5}$$

We can now express the product of f with g as

$$fg = \hat{H}FG, \tag{2.6}$$

where $\deg F, \deg G < d$. Thus the polynomial multiplication can be done through a multiplication of a vector with a Toeplitz matrix and vector dot product. We shall develop this idea further to compute $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$. First we recall that we can also go in the other direction, i.e., reduce a multiplication of a vector with a Toeplitz matrix to multiplication of polynomials.

MiddleProduct

Algorithm `MiddleProduct` in Figure 2.1 is used to the compute the product of a polynomial vector with a Toeplitz matrix by reducing it to polynomial multiplication. Algorithm `MiddleProduct` in Figure 2.1 takes as input $H \in \mathsf{K}[x]^{1\times\ell}$ and $f \in \mathsf{K}[x]$ and outputs HF, where F is a Toeplitz matrix as described in the algorithm.

`MiddleProduct`(f, H, ℓ, m, d)
Input: $H \in \mathsf{K}[x]^{1\times\ell}, f \in \mathsf{K}[x], m, d \in \mathbb{Z}_{>0}$
Condition: $\deg f < (l - m + 1)d$.
Let $t = \ell - m$, $u = m - 1$, write f as $f = f_0 + f_1 x^d + \cdots + f_t x^{td}$

Let $F = \begin{bmatrix} f_0 & & & \\ f_1 & f_0 & & \\ & f_1 & \ddots & \\ \vdots & \vdots & & f_0 \\ & & & f_1 \\ f_t & & & \\ & f_t & & \vdots \\ & & \ddots & \\ & & & f_t \end{bmatrix} \in \mathsf{K}[x]^{\ell\times m}$

Output: $HF \in \mathsf{K}[x]^{1\times m}$

$\gamma = \deg(H) + d$;
$f_{\exp} := f_t + f_{t-1}x^\gamma + f_{t-2}x^{2\gamma} + \cdots + f_0 x^{t\gamma}$;
$h_{\exp} := (H)_1 + (H)_2 x^\gamma + (H)_3 x^{2\gamma} + \cdots + (H)_\ell x^{(l-1)\gamma}$;
$a_{\exp} := f_{\exp} \times h_{\exp}$;
Write $a_{\exp}$ as $a_{\exp} = a_0 + a_1 x^\gamma + a_2 x^{2\gamma} + \cdots + a_{t+l-1} x^{(t+l-1)\gamma}$
return $\begin{bmatrix} a_t & a_{t+2} & \cdots & a_{t+u} \end{bmatrix}$

Figure 2.1: Algorithm `MiddleProduct`

First, the algorithm augments the matrix F to

$$F_{\text{aug}} = \left[\begin{array}{cccc|cccccc|cccc} f_t & \cdots & & f_1 & f_0 & & & & & & & & & \\ & f_t & \cdots & & f_1 & f_0 & & & & & & & & \\ & & f_t & \cdots & f_0 & f_1 & \ddots & & & & & & & \\ & & & \ddots & \vdots & \vdots & & f_0 & & & & & & \\ & & & & & & & f_1 & f_0 & & & & & \\ & & & & f_t & & & & f_1 & & f_0 & & & \\ & & & & & f_t & & \vdots & \vdots & & f_1 & f_0 & & \\ & & & & & & \ddots & & & & \vdots & & \ddots & \\ & & & & & & & f_t & & & & & & \\ & & & & & & & & f_t & & f_{t-1} & \cdots & f_1 & f_0 \end{array}\right] \in \mathsf{K}[x]^{\ell\times(m+2t)}$$

We now compute HF_{aug}, instead of HF, as we can compute HF_{aug} using polynomial multiplication of

$$f_{\text{exp}} = f_t + f_{t-1}y + f_{t-2}y^2 + \cdots + f_0 y^t \in \mathsf{K}[x][y]$$

with

$$h_{\text{exp}} = (H)_1 + (H)_2 y + (H)_3 y^2 + \cdots + (H)_\ell y^{(l-1)}.$$

Next, we use Kronecker substitution, $y = x^\gamma$, to obtain the same result in x^γ-adic expansion, where $\deg_x f_{\text{exp}} + \deg_x h_{\text{exp}} < \gamma = \deg(H) + d$. This shift from x^d-adic system to x^γ-adic system ensures a one to one correspondence between the x^γ-adic coefficients of the product and the entries of the vector HF_{aug}. Finally, we multiply

$$f_{\text{exp}} = f_t + f_{t-1}x^\gamma + f_{t-2}x^{2\gamma} + \cdots + f_0 x^{t\gamma}$$

with

$$h_{\text{exp}} = (H)_1 + (H)_2 x^\gamma + (H)_3 x^{2\gamma} + \cdots + (H)_\ell x^{(l-1)\gamma}.$$

The coefficients in the middle of the polynomial give the entries of the vector HF.

Algorithm `MiddleProduct`(f, H, ℓ, m, d) in Figure 2.1 involves only a single multiplication of two univariate polynomials of degree less than $l \cdot (\deg H + d)$. We obtain the following result.

Lemma 2.2. *Algorithm* `MiddleProduct` *in Figure 2.1 is correct and costs* $\mathsf{M}(l \cdot (\deg H + d))$ *basic operations from* K.

Here is an illustration of Algorithm `MiddleProduct` in Figure 2.1.

Example 2.3. *Let us work in the field* $\mathbb{Z}_7$. *Let*

$$\begin{aligned} H &= \begin{bmatrix} 6x^3+6x^2+4x+1 & 4x^2+5x+4 & 3x^3+6x^2+4x+3 & 4x+4 \end{bmatrix} \\ f &= 2x^3+4x^2+4x+2 \\ d &= 2 \\ F &= \begin{bmatrix} 4x+2 & & \\ 2x+4 & 4x+2 & \\ 0 & 2x+4 & \\ & 0 & \end{bmatrix} \end{aligned}$$

To compute HF, we shall augment the matrix F to

$$F_{aug} = \left[\begin{array}{cc|cc|cc} 0 & 2x+4 & 4x+2 & & & \\ & 0 & 2x+4 & 4x+2 & & \\ & & 0 & 2x+4 & 4x+2 & \\ & & & 0 & 2x+4 & 4x+2 \end{array}\right].$$

So we get

$$\begin{aligned} \gamma =& 5, \\ f_{exp} =& 4x^{11} + 2x^{10} + 2x^6 + 4x^5, \\ h_{exp} =& 4x^{16} + 4x^{15} + 3x^{13} + 6x^{12} + 4x^{11} + 3x^{10} + 4x^7 + 5x^6 + 4x^5 + 6x^3 + 6x^2 + 4x + 1, \end{aligned}$$

and

$$\begin{aligned} h_{exp} f_{exp} =& (2x^2 + 3x + 1) \times x^{25} + (5x^4 + 2x^3 + x^2 + 2x + 1) \times x^{20} + (6x^4 + 5x^3 + 4x^2 + 6x + 6) \times x^{15} \\ &+ (3x^4 + 2x^3 + 5x^2 + 5x + 4) \times x^{10} + (5x^4 + x^3 + 4x^2 + 4x + 4) \times x^5. \end{aligned}$$

The part corresponding to HF would be coefficient of x^{10} and x^{15}, i.e.,

$$[3x^4 + 2x^3 + 5x^2 + 5x + 4 \quad 6x^4 + 5x^3 + 4x^2 + 6x + 6].$$

PartialProduct

Now we turn to the main problem of this section: calculate $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$. Algorithm `PartialProduct`$(f, g, s, d)$ in Figure 2.2 calculates $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$, where $f, g, s \in \mathsf{K}[x]$ and $d \in \mathbb{Z}_{>0}$. As we calculate only the low order coefficients of the product $\mathrm{Rem}(fg, s)$, we call this algorithm "PartialProduct." In the next section we shall scale this method to handle vector outer products. The cost of the `PartialProduct`(f, g, s, d) method, for two polynomials, is $O((\deg s)/d \cdot \mathsf{M}(\deg s))$. This cost is much higher than that of the standard method; it is only presented to help understand the case where single polynomials are replaced by column and row vectors.

Consider the problem of multiplication of two polynomials $f, g \in \mathsf{K}[x]$. Using equations (2.1) − (2.5), we get

$$\begin{aligned} fg &= \hat{H}FG, \\ \mathrm{Rem}(\mathrm{Rem}(fg, s), x^d) &= \mathrm{Rem}(\mathrm{Rem}(\hat{H}FG), x^d) \\ &= \mathrm{Rem}(\mathrm{Rem}(\mathrm{Rem}(\hat{H}, s)FG, s), x^d). \end{aligned}$$

Let $H = \mathrm{Rem}(\hat{H}, s)$ and $q = \mathrm{Quo}(HFG, s)$, we get

$$\begin{aligned}\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d) &= \mathrm{Rem}(\mathrm{Rem}(HFG, s), x^d) \\ &= \mathrm{Rem}(HFG - qs, x^d) \\ &= \mathrm{Rem}(HFG, x^d) - \mathrm{Rem}(qs, x^d) \\ &= \mathrm{Rem}(HFG, x^d) - \mathrm{Rem}(q\mathrm{Rem}(s, x^d), x^d). \end{aligned} \tag{2.7}$$

It is given that

$$\begin{aligned}\deg F &< d, \\ \deg G &< d, \\ \deg H &< \deg s \text{ (by assumption)},\end{aligned}$$

thus,

$$\begin{aligned}\deg HFG &< \deg s + 2d, \\ \deg q &< 2d.\end{aligned}$$

Let $z = \lceil \deg s/d \rceil$. We split $H = H^{(l)} + H^{(u)}$, where $H^{(l)}$ consists of the low order terms of H :.

$$\begin{aligned}H^{(l)} &= \mathrm{Rem}(H, x^{d(z-2)}), \\ H^{(u)} &= H - H^{(l)}, \\ HFG &= H^{(l)}FG + H^{(u)}FG.\end{aligned}$$

Since $\deg(H^{(l)}FG) < \deg s$, $H^{(l)}FG$ does not contribute to $\mathrm{Quo}(HFG)$.

$$\begin{aligned}\mathrm{Quo}(HFG, s) &= \mathrm{Quo}(H^{(u)}FG, s) \\ &= \mathrm{Quo}(x^{d(z-2)}\mathrm{Quo}(H, x^{d(z-2)})FG, s). \end{aligned} \tag{2.8}$$

Let $X = x^d$, putting (2.7) and (2.8) together we get

$$\begin{aligned}\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d) =& \mathrm{Rem}(HFG, x^d) - \mathrm{Rem}(q\mathrm{Rem}(s, x^d), x^d) \\ =& \mathrm{Rem}(\mathrm{Rem}(H, X)FG, X) \\ & - \mathrm{Rem}(\mathrm{Quo}(X^{z-2}\mathrm{Quo}(H, X^{(z-2)})FG, s)\mathrm{Rem}(s, X), X).\end{aligned} \tag{2.9}$$

Equation (2.9) summarizes the whole Algorithm 2.2. We next analyze the cost of Algorithm 2.2.

The cost of Preprocessing phase of Algorithm `PartialProduct` in figure 2.2 is bounded by the cost of computing H, i.e., $O((u+t)\cdot(\mathsf{M}(\deg s)+\mathsf{M}(d)))$ basic operations from K. Using the fact that $u, t \in O((\deg s)/d)$, we can bound the cost of phase 1 by $O((\deg s)/d \cdot \mathsf{M}(\deg s))$. The cost of MiddleProduct phase is bounded by $O(\mathsf{M}((u+t)d))$, using lemma 2.2. Using the bounds on u and v, this cost reduces to $O(\mathsf{M}(\deg s))$. The cost of Vector dot product phase is bounded by $O(u\mathsf{M}(d))$, which reduces to $O((\deg s)/d\mathsf{M}(d))$. Putting everything together to get the final result in phase 4 costs $O(\mathsf{M}(d))$ basic operations from K. Therefore, the cost of the algorithm is dominated by the first phase. We can now state the following result.

Lemma 2.4. *Algorithm* `PartialProduct`(f, g, s, d) *in Figure 2.2, is correct and costs*

$$O((\deg s)/d\ (\mathsf{M}(\deg s) + \mathsf{M}(d)))$$

basic field operations from K.

Let us illustrate Algorithm 2.2 with the following example.

Example 2.5. *Let us work in* $\mathbb{Z}_7$. *We want to find* $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$, *where*

$f = 4x^5+6x^3+3x^2+3x+4,\ g = 2x^5+x^4+6x^3+x+3,\ s = 4x^6+2x^5+3x^4+6x^2$ *and* $d = 2$.

In Preprocessing phase , we compute H *and trim it to the required low order and high order parts* $H^{(l)}$ *and* $\hat{H}$.

$$\begin{aligned}
H =&[1 \quad x^2 \quad x^4 \quad 3x^5 + x^4 + 2x^2 \quad 5x^5 + 5x^4 + 6x^3 + 6x^2 \quad x^5 + 5x^4 + 3x^3 + 5x^2]\\
H^{(l)} =&[1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]\\
H^{(u)} =&[0 \quad 1 \quad x^2 \quad 3x^3 + x^2 + 2 \quad 5x^3 + 5x^2 + 6x + 6 \quad x^3 + 5x^2 + 3x + 5].
\end{aligned}$$

In MiddleProduct phase we compute the middle product of $H^{(l)}$ *and* $H^{(u)}$ *with* f *to get* $F^{(l)}$ *and* $F^{(u)}$ *using algorithm* 2.1.

$$\begin{aligned}
F^{(l)} =&[3x + 4 \quad 0 \quad 0]\\
F^{(u)} =&[4x^3 + 6x + 3 \quad 5x^4 + 3x^3 + 3x^2 + 4x + 4 \quad 3x^4 + 3x^3 + 3x^2 + x + 6].
\end{aligned}$$

In Vector dot product phase, we find the dot product of $F^{(l)}$ *with* G *and* $F^{(u)}$ *with* G.

$$\begin{aligned}
p^{(l)} =&3x^2 + 6x + 5,\\
p^{(u)} =&x^5 + 3x^4 + 4x^3 + 2x + 1
\end{aligned}$$

where

$$G = [x + 3 \quad 6x \quad 2x + 1]^T.$$

In phase 4, *we compute* $q = 2x + 5$ *and finally put together everything to get the result*

$$\mathrm{Rem}(3x^2 + 6x + 5, x^2) - \mathrm{Rem}((2x + 5)\mathrm{Rem}(4x^6 + 2x^5 + 3x^4 + 6x^2, x^2), x^2) = 6x + 5.$$

`PartialProduct`(f, g, s, d)
Input: $s, f, g \in \mathsf{K}[x], d \in \mathbb{Z}_{>0}$
Condition: $s \neq 0,\ f = \mathrm{Rem}(f, s),\ g = \mathrm{Rem}(g, s)$
Output: $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$

if $\deg(s) \leq 2d$
  return $\mathrm{Rem}(\mathrm{Rem}(fg, s), x^d)$
fi
$t, u, z, X = \lfloor \frac{\deg(f)}{d} \rfloor, \lfloor \frac{\deg(g)}{d} \rfloor, \lfloor \frac{\deg(s)}{d} \rfloor, x^d;$

[1: Preprocessing]
$\hat{H} = [\ 1 \quad X \quad X^2 \quad \cdots \quad X^{t+u+1}\ ];$
$H := \mathrm{Rem}(\hat{H}, s);$
$H^{(l)} := \mathrm{Rem}(H, X)\ ,\ H^{(u)} := \mathrm{Quo}(H, X^{z-2});$

[2: MiddleProduct]
$F^{(l)} := \mathrm{MiddleProduct}(f, H^{(l)}, u + t + 1, u + 1, d) \in K[x]^{1 \times u+1};$
$F^{(u)} := \mathrm{MiddleProduct}(f, H^{(u)}, u + t + 1, u + 1, d) \in K[x]^{1 \times u+1};$

[3: Vector dot product]
Write g as: $g = g_0 + g_1 X + g_2 X^2 + \cdots + g_u X^u$
$G = \left[\ g_0 \quad g_1 \quad g_2 \quad \cdots \quad g_u\ \right]^T;$
$p^{(l)}, p^{(u)} := F^{(l)}.G, F^{(u)}.G;$

[4:Combining the terms]
$q := \mathrm{Quo}(x^{d(z-2)} p^{(u)}, s)$
return $\mathrm{Rem}(p^{(l)}, X) - \mathrm{Rem}(q \times \mathrm{Rem}(S, X), X)$

Figure 2.2: Algorithm `PartialProduct`(f, g, s, d)

2.2 Vector Case: Computing $\mathrm{Rem}(\mathrm{Rem}(FG, s), x^d)$

Let $F \in \mathsf{K}[x]^{n\times 1}, G \in \mathsf{K}[x]^{1\times n}$, and $\deg F, \deg G < \deg s$. This section deals with the problem of computing $\mathrm{Rem}(\mathrm{Rem}(FG, s), x^d)$. Algorithm `PartialProduct`$(F, G, s, d)$ in Figure 2.3 is analogous to Algorithm 2.2, with polynomials f and g replaced by vectors of polynomials F and G.

The preprocessing phase in Algorithm 2.3 is same as that in Algorithm 2.2 and costs $O((\deg s)/d \cdot \mathsf{M}(\deg s))$ operations from K. MiddleProduct phase is also same as that in algorithm 2.2 except that we repeat the procedure for every entry in F. The cost of this phase is bounded by $O(n\mathsf{M}(\deg s))$ operations from K (Lemma 2.2). The MatrixMultiplication phase in Algorithm 2.3 involves matrix multiplication instead of dot product and costs $O(\mathsf{MM}(n, (\deg s)/d, n, d))$ operations from K. The last phase puts everything together and costs $O(\deg s + n^2\,\mathsf{M}(d))$ operations from K.

The cost of Algorithm 2.2 was dominated by the preprocessing phase. The cost of Algorithm 2.3 is dominated by the matrix multiplication step in Phase 3. This is due to the fact that the cost of the preprocessing phase is not related to the number of polynomials but the degree bounds on the polynomials. Thus as the number of terms increase the cost of Phase 3 dominates the cost of the algorithm. Now we can state the following result:

Theorem 2.6. *The algorithm* 2.3*, PartialProduct, is correct and costs*

$$O(\mathsf{MM}(n, (\deg s)/d, n, d))$$

operations from K.

Let us illustrate Algorithm 2.3 with the following example.

Example 2.7. *Let us work in* $\mathbb{Z}_7$*. Let*

$$\begin{aligned} F =& [0 \quad x^3 + x^2 + 5x + 6 \quad 6x + 1]^T, \\ G =& [0 \quad 6x^4 + 6x^3 + 2x^2 + 4x + 1 \quad 4x^5 + 5x^4 + 3x^3 + 3x^2 + 2], \\ s =& 6x^6 + 6x^5 + x^4 + 6x^3 + 4x^2 + 3, \ \textit{and } d = 2. \end{aligned}$$

In Preprocessing phase, we compute H *and trim it to low order and high order parts* $H^{(l)}$ *and* $H^{(u)}$*, required for the computation.*

$$\begin{aligned} H =& [1 \quad x^2 \quad x^4 \quad 6x^5 + x^4 + 6x^3 + 4x^2 + 3 \quad 3x^5 + x^3 + 4x^2 + 4x + 6], \\ H^{(l)} =& [1 \quad 0 \quad 0 \quad 3 \quad 4x + 6], \\ H^{(u)} =& [1 \quad 0 \quad x^2 \quad 6x^3 + x^2 + 6x + 4 \quad 3x^3 + x + 4]. \end{aligned}$$

`PartialProduct`(F, G, s, d)
Input: $s \in \mathsf{K}[x], F \in \mathsf{K}[x]^{n\times 1}, G \in \mathsf{K}[x]^{1\times n}, d \in \mathbb{Z}_{>0}$
Condition: $s \neq 0$, $G = \mathrm{Rem}(G, s)$, $F = \mathrm{Rem}(F, s)$
Output: $\mathrm{Rem}(\mathrm{Rem}(FG, s), x^d)$

if $\deg(s) \leq 2d$
  return $\mathrm{Rem}(\mathrm{Rem}(FG, s), x^d)$;
fi
$t, u, z, X := \lfloor \frac{\deg(F)}{d} \rfloor, \lfloor \frac{\deg(G)}{d} \rfloor, \lfloor \frac{\deg(z)}{d} \rfloor, x^d$;

[1: Preprocessing]
$\hat{H} := [\ 1 \quad X \quad X^2 \quad \cdots \quad X^{t+u+1}\ ]$;
$H := \mathrm{Rem}(\hat{H}, s)$;
$H^{(l)} := \mathrm{Rem}(H, X)$, $H^{(u)} := \mathrm{Quo}(H, X^{z-1})$;

[2: MiddleProduct]
for $i = 1$ *to* n
  $F_i^{(l)} := \mathrm{MiddleProduct}(F_i, H^{(l)}, u+t+1, u+1, d) \in K[x]^{1\times u+1}$;
  $F_i^{(u)} := \mathrm{MiddleProduct}(F_i, H^{(u)}, u+t+1, u+1, d) \in K[x]^{1\times u+1}$;
od

[Phase 3: Matrix multiplication]
Write G_i as: $(G)_i = g_{i,0} + g_{i,1}X + g_{i,2}X^2 + \cdots + g_{i,u}X^u$ for $1 \leq j \leq n$.

$$G = \begin{bmatrix} g_{1,0} & g_{2,0} & g_{3,0} & \cdots & g_{n,0} \\ g_{1,1} & g_{2,1} & g_{3,1} & \cdots & g_{n,1} \\ g_{1,2} & g_{2,2} & g_{3,2} & \cdots & g_{n,2} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ g_{1,u} & g_{2,u} & g_{3,u} & \cdots & g_{n,u} \end{bmatrix} \in \mathsf{K}[x]^{(u+1)\times n}.$$

$$F_M^{(l)}, F_M^{(u)} = \left[\begin{array}{c} F_{L1} \\ \hline F_{L2} \\ \hline F_{L3} \\ \hline \vdots \\ \hline F_{Ln} \end{array}\right], \left[\begin{array}{c} F_{U1} \\ \hline F_{U2} \\ \hline F_{U3} \\ \hline \vdots \\ \hline F_{Un} \end{array}\right] \in \mathsf{K}[x]^{n\times(u+1)}.$$

$P^{(l)}, P^{(u)} := F_M^{(l)}.G, F_M^{(u)}.G$

[Phase 4: Combining the terms]
$Q := \mathrm{Quo}(x^{d(z-2)}\, P^{(u)}, s)$
return $\mathrm{Rem}(P^{(l)}, X) - \mathrm{Rem}(\mathrm{Rem}(Q, X) \cdot \mathrm{Rem}(s, X), X)$;

Figure 2.3: Algorithm `PartialProduct`(F, G, s, d)

In MiddleProduct phase, we compute the middle product of $H^{(l)}$ and $H^{(u)}$ with each entry in F to get $F_M^{(l)}$ and $F_M^{(u)}$ using Algorithm 2.1.

$$F_M^{(l)} = \begin{bmatrix} 0 & 0 & 0 \\ 5x+6 & 0 & 3x+3 \\ 6x+1 & 0 & 0 \end{bmatrix}$$
$$F_M^{(l)} = \begin{bmatrix} 0 & 0 & 0 \\ x+1 & x^3+x^2+5x+6 & 6x^4+5x^3+6x^2+3x+4 \\ 0 & 6x+1 & 6x^3+x^2 \end{bmatrix}.$$

We then multiply the matrices $F_M^{(l)}$ and $F_M^{(u)}$ with G to get $P^{(l)}$ and $P^{(u)}$ in MatrixMultiplication phase:

$$G = \begin{bmatrix} 0 & 4x+1 & 2 \\ 0 & 6x+2 & 3x+3 \\ 0 & 6 & 4x+5 \end{bmatrix}$$
$$P^{(l)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 6x^2+5x+3 & 5x^2+2x+6 \\ 0 & 3x^2+3x+1 & 5x+2 \end{bmatrix}$$
$$P^{(u)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3x^3+2x^2+6x+2 & 3x^5+4x^4+6x^3+4x^2+3x+5 \\ 0 & x^3+4x+2 & 3x^4+6x^3+2x^2+3 \end{bmatrix}.$$

Finally we find the quotient and put everything together in phase 4.

$$\begin{aligned} Q =& Quo(x^2\, P^{(u)}, s) \\ =& \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 4x+6 \\ 0 & 0 & 4 \end{bmatrix}. \end{aligned}$$

Thus the output is

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 5x+3 & 4x+2 \\ 0 & 3x+1 & 5x+4 \end{bmatrix}.$$

2.3 Array Case: Computing $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d)$

We now extend the `PartialProduct` algorithm to compute $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d)$, where $A \in \mathsf{K}[x]^{n\times\ell}$, $B \in \mathsf{K}[x]^{\ell\times m}$, $h, s \in \mathsf{K}[x]$ and $d \in \mathbb{Z}_{>0}$, along with the conditions that

$\deg A, \deg B, \deg h < \deg s$. The problem in this section is more general than the problems tackled in the previous sections, but the central idea remains the same. To keep the description of the main idea simple, we chose to solve a slightly simplified problem in the previous sections.

Algorithm `PartialProduct`(h, A, B, s, d) in Figure 2.4 describes the procedure to compute $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d)$. It is exactly analogous to algorithm 2.2 and algorithm 2.3, except for phase 1 which accommodates for the multiplication of h by AB, (The same change would be required for computation of $\mathrm{Rem}(\mathrm{Rem}(hfg, s), x^d)$ by Algorithm 2.2. This can be seen easily by multiplying both sides of (2.6) by h.)

We now discuss the cost of Algorithm 2.4. The cost of the preprocessing phase is bounded by $O((u+t) \cdot \mathsf{M}(\deg s))$. Using lemma 2.2, we can bound the cost of the Middle Product phase by $O(nl\mathsf{M}((u+t)d))$. The matrix multiplication phase has cost bounded by $O(\mathsf{MM}(n, (u+1)l, m, d))$ and the cost of the last phase is bounded by $O(nm\mathsf{M}(d))$. Thus we can state the following result.

Lemma 2.8. *The cost of Algorithm* `PartialProduct`(h, A, B, s, d) *in Figure 2.4 is bounded by* $O(1/d\,(\deg A + \deg B)\mathsf{M}(\deg s + d) + nl\mathsf{M}(\deg A + \deg B) + \mathsf{MM}(n, \ell \deg B/d, m, d))$ *basic operations from* K. *Here* $A \in \mathsf{K}[x]^{n\times\ell}$ *and* $B \in \mathsf{K}[x]^{\ell\times m}$.

We also like to mention the following corollary, which will be useful in the next chapter.

Corollary 2.9. *If* $A \in \mathsf{K}[x]^{n\times\ell}$, $B \in \mathsf{K}[x]^{\ell\times m}$, $l \deg B, \ell \deg A \in O(nd)$ *and* $\deg s \leq nd$, *then the cost of Algorithm* `PartialProduct`(h, A, B, s, d) *in Figure 2.4 is bounded by* $O(n\mathsf{M}(nd) + n^\omega\,\mathsf{M}(d))$ *basic operations from* K.

2.4 Extension: Compute $\mathrm{Quo}(\mathrm{Rem}(\mathrm{Rem}(hAB, s), X^{k+1}), X^k)$

We can easily extend Algorithm 2.4 to find any x^d-adic coefficient of $\mathrm{Rem}(hAB, s)$ in the same cost. For the sake of keeping the presentation simple and the use of this algorithm in the rest of the paper, we have described the algorithm only for the trailing coefficient case, in the previous sections.

Let $X = x^d$. The basis of all `PartialProduct` algorithms described in previous sections has been (2.9). We shall now derive a similar equation for the case of finding the X adic coefficient of X^k in $\mathrm{Rem}(fg, s)$, i.e.,

$$\mathrm{Quo}(\mathrm{Rem}(\mathrm{Rem}(fg, s), X^{k+1}), X^k).$$

`PartialProduct`(h, A, B, s, d)
Input: $A \in K[x]^{n\times l}, B \in K[x]^{l\times m}, s, h \in K[x], d \in \mathbb{Z}_{>0}$
Condition: $s \neq 0$, $h = \mathrm{Rem}(h, s)$, $A = \mathrm{Rem}(A, s)$ and $B = \mathrm{Rem}(B, s)$
Output: $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d) \in \mathsf{K}[x]^{n\times m}$

if $\deg(s) \leq 2d$
  return $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d)$;
fi
$u, t, z, X := \lfloor \frac{\deg(B)}{d} \rfloor, max\{\lfloor \frac{\deg(A)}{d} \rfloor, u\}, \lfloor \frac{\deg(s)}{d} \rfloor, x^d$;

[1 : Preprocessing]
$\hat{H} = [\ h \quad hX \quad hX^2 \quad \cdots \quad hX^{t+u+1}\ ]$;
$H := \mathrm{Rem}(\hat{H}, s)$;
$H^{(l)}, H^{(u)} := \mathrm{Rem}(H, X), \mathrm{Quo}(H, X^{z-1})$;

[2: MiddleProduct]
for each $f_{ij} = A_{ij}$
  $F_{ij}^{(l)} := \mathrm{MiddleProduct}(f_{ij}, H^{(l)}, u+t+1, u+1, d) \in K[x]^{1\times u+1}$;
  $F_{ij}^{(u)} := \mathrm{MiddleProduct}(f_{ij}, H^{(u)}, u+t+1, u+1, d) \in K[x]^{1\times u+1}$
od

[3: Matrix multiplication]
for $i = 1$ *to* l
  Write $B_{ij} = g_{j,0} + g_{j,1}X + \cdots + g_{j,u}X^u$, for $1 \leq j \leq m$.

$$G_i = \begin{bmatrix} g_{1,0} & g_{2,0} & \cdots & g_{m,0} \\ g_{1,1} & g_{2,1} & \cdots & g_{m,1} \\ \vdots & \vdots & \cdots & \vdots \\ g_{1,u} & g_{2,u} & \cdots & g_{m,u} \end{bmatrix} \in \mathsf{K}[x]^{(u+1)\times m}$$

od

$$G = \left[\begin{array}{c} G_1 \\ \hline \\ \vdots \\ \hline \\ G_{l-1} \\ \hline \\ G_l \end{array}\right] \in \mathsf{K}[x]^{(u+1)l\times m},$$

$$F_M^{(l)}, F_M^{(u)} = \begin{bmatrix} F_{11}^{(l)} & F_{12}^{(l)} & \cdots & F_{1(l-1)}^{(l)} & F_{1l}^{(l)} \\ F_{21}^{(l)} & F_{22}^{(l)} & \cdots & F_{2(l-1)}^{(l)} & F_{2l}^{(l)} \\ \vdots & \vdots & & \vdots & \vdots \\ F_{n1}^{(l)} & F_{n2}^{(l)} & \cdots & F_{n(l-1)}^{(l)} & F_{nl}^{(l)} \end{bmatrix}, \begin{bmatrix} F_{11}^{(u)} & F_{12}^{(u)} & \cdots & F_{1(l-1)}^{(u)} & F_{1l}^{(u)} \\ F_{21}^{(u)} & F_{22}^{(u)} & \cdots & F_{2(l-1)}^{(u)} & F_{2l}^{(u)} \\ \vdots & \vdots & & \vdots & \vdots \\ F_{n1}^{(u)} & F_{n2}^{(u)} & \cdots & F_{n(l-1)}^{(u)} & F_{nl}^{(u)} \end{bmatrix}$$

$P^{(l)}, P^{(u)} := F_{\mathrm{M}}^{(l)}.G_M, F_{\mathrm{M}}^{(u)}.G_M \in \mathsf{K}[x]^{n\times m}$;

[4: Combining the terms]
$Q := \mathrm{Quo}(X^{z-2}P^{(u)}, s)$
return $\mathrm{Rem}(P^{(l)}, X) - \mathrm{Rem}(\mathrm{Rem}(Q, X) \times \mathrm{Rem}(s, X), X)$;

Figure 2.4: Algorithm `PartialProduct`(h, A, B, s, d)

We use (2.1) − (2.5) to get

$$\begin{aligned} fg &= \hat{H}FG \\ \Rightarrow \quad \mathrm{Rem}(\mathrm{Rem}(fg, s), X^{k+1}) &= \mathrm{Rem}(\mathrm{Rem}(\hat{H}FG), X^{k+1}) \\ &= \mathrm{Rem}(\mathrm{Rem}(\mathrm{Rem}(\hat{H}, s)FG, s), X^{k+1}). \end{aligned}$$

Let $H = \mathrm{Rem}(\hat{H}, s)$ and $q = \mathrm{Quo}(HFG, s)$, now the above equation can be written as

$$\begin{aligned} \mathrm{Rem}(\mathrm{Rem}(fg, s), X^{k+1}) &= \mathrm{Rem}(\mathrm{Rem}(HFG, s), X^{k+1}) \\ &= \mathrm{Rem}(HFG - qs, X^{k+1}) \\ &= \mathrm{Rem}(HFG, X^{k+1}) - \mathrm{Rem}(qs, X^{k+1}) \\ &= \mathrm{Rem}(HFG, X^{k+1}) - \mathrm{Rem}(q\mathrm{Rem}(s, X^{k+1}), X^{k+1}) \\ &= \mathrm{Rem}(H_{k+1}FG, X^{k+1}) \\ &\qquad - \mathrm{Rem}(q\mathrm{Rem}(s, X^{k+1}), X^{k+1}), \end{aligned} \tag{2.10}$$

where $H_{k+1} = \mathrm{Rem}(H, X^{k+1})$. It is given that

$$\begin{aligned} \deg F &< d, \\ \deg G &< d, \text{ and} \\ \deg H &< \deg s \text{ (by definition)}, \end{aligned}$$

thus,

$$\begin{aligned} \deg HFG &< \deg s + 2d, \text{ and} \\ \deg q &< 2d. \end{aligned}$$

Let $z = \lceil \deg s/d \rceil$. We split $H = H^{(l)} + H^{(u)}$, where $H^{(l)}$ contains the lower order terms of H, in the following manner:

$$\begin{aligned} H^{(l)} &= \mathrm{Rem}(H, X^{z-2}), \\ H^{(u)} &= H - H^{(l)} \\ \Rightarrow \quad HFG &= H^{(l)}FG + H^{(u)}FG. \end{aligned}$$

Since $\deg(H^{(l)}FG) < \deg s$

$$\begin{aligned} \mathrm{Quo}(HFG, s) &= \mathrm{Quo}(H^{(u)}FG, s) \\ &= \mathrm{Quo}(X^{z-2}\mathrm{Quo}(H, X^{z-2})FG, s). \end{aligned} \tag{2.11}$$

Putting (2.10) and (2.11) together we get as before (2.9)

$$\begin{aligned}\mathrm{Rem}(\mathrm{Rem}(fg,s),X^{k+1}) =& \mathrm{Rem}(H_{k+1}FG,X^{k+1}) - \mathrm{Rem}(q\mathrm{Rem}(s,X^{k+1}),X^{k+1}) \qquad (2.12)\\ =& \mathrm{Rem}(H_{k+1}FG,X^{k+1})\\ & - \mathrm{Rem}(\mathrm{Quo}(X^{z-2}\mathrm{Quo}(H,X^{(z-2)})FG,s)\mathrm{Rem}(s,X^{k+1}),X^{k+1}).\end{aligned}$$

We now split H_{k+1} as

$$H_{k+1} = H_{k+1}^{(1)} + X^{k-2}H_{k+1}^{(2)},$$

where

$$H_{k+1}^{(1)} = \mathrm{Rem}(H,X^{k-2}).$$

Therefore

$$\begin{aligned} H_{k+1}FG &= H_{k+1}^{(1)}FG + X^{(k-2)}H_{k+1}^{(2)}FG\\ \Rightarrow \quad \mathrm{Rem}(H_{k+1}FG,X^{k+1}) &= \mathrm{Rem}(H_{k+1}^{(1)}FG,X^{k+1})\\ &\quad + \mathrm{Rem}(X^{(k-2)}H_{k+1}^{(2)}FG,X^{k+1}). \qquad (2.13)\end{aligned}$$

Since the degree of the first term in (2.13) is less than kd, we get

$$\mathrm{Quo}(\mathrm{Rem}(H_{k+1}FG,X^{k+1}),X^k) = \mathrm{Quo}(\mathrm{Rem}(X^{k-2}H_{k+1}^{(2)}FG,X^{k+1}),X^k). \qquad (2.14)$$

Similarly let $\mathrm{Rem}(s,X^{k+1}) = s_{k+1}$ and let $s_{k+1} = s_{k+1}^{(1)} + X^{k-2}s_{k+1}^{(2)}$. We get

$$\mathrm{Quo}(\mathrm{Rem}(q\mathrm{Rem}(s,X^{k+1}),X^{k+1}),X^k) = \mathrm{Quo}(\mathrm{Rem}(qX^{k-2}s_{k+1}^{(2)},X^{k+1}),X^k). \qquad (2.15)$$

Using (2.12), (2.14) and (2.15) together we get

$$\begin{aligned}\mathrm{Quo}(\mathrm{Rem}(\mathrm{Rem}(fg,s),X^{k+1}),X^k) = & \quad \mathrm{Quo}(\mathrm{Rem}(X^{k-2}H_{k+1}^{(2)}FG,X^{k+1}),X^k)\\ & - \mathrm{Quo}(\mathrm{Rem}(qX^{k-2}s_{k+1}^{(2)},X^{k+1}),X^k), \qquad (2.16)\end{aligned}$$

where

$$\begin{aligned} H &= \mathrm{Rem}(\hat{H},s),\\ H_{k+1}^{(2)} &= \mathrm{Quo}(\mathrm{Rem}(H,X^{k+1}),X^k),\\ s_{k+1}^{(2)} &= \mathrm{Quo}(\mathrm{Rem}(s,X^{k+1}),X^k), \text{ and}\\ q &= \mathrm{Quo}(\mathrm{Quo}(H,X^{z-2})FG,s).\end{aligned}$$

`PartialProduct`(f, g, s, d, k)
Input: $s, f, g \in \mathsf{K}[x], d, k \in \mathbb{Z}_{>0}$
Condition: $s \neq 0,\ f = \mathrm{Rem}(f, s),\ g = \mathrm{Rem}(g, s),\ kd < \deg s$
Let $X = x^d$.
Output: $\mathrm{Quo}(\mathrm{Rem}(\mathrm{Rem}(fg, s), X^{k+1}), X^k)$

if $\deg(s) \leq 2d$
  return $\mathrm{Quo}(\mathrm{Rem}(\mathrm{Rem}(fg, s), X^{k+1}), X^k)$
fi
$t, u, z = \lfloor \frac{\deg(f)}{d} \rfloor, \lfloor \frac{\deg(g)}{d} \rfloor, \lfloor \frac{\deg(s)}{d} \rfloor$;

[1: Preprocessing]
$\hat{H} = [\ 1 \quad X \quad X^2 \quad \cdots \quad X^{t+u+1}\ ]$;
$H := \mathrm{Rem}(\hat{H}, s)$;
$H^{(l)}, H^{(u)} := \mathrm{Quo}(\mathrm{Rem}(H, X^{k+1}), X^k), \mathrm{Quo}(H, X^{z-2})$;
$s_l = \mathrm{Quo}(\mathrm{Rem}(s, X^{k+1}), X^k)$

[2: MiddleProduct]
$F^{(l)} := \mathrm{MiddleProduct}(f, H^{(l)}, u + t + 1, u + 1, d) \in K[x]^{1 \times u+1}$;
$F^{(u)} := \mathrm{MiddleProduct}(f, H^{(u)}, u + t + 1, u + 1, d) \in K[x]^{1 \times u+1}$;

[3: Vector dot product]
Write g as: $g = g_0 + g_1 X + g_2 X^2 + \cdots + g_u X^u$
$G = [\ g_0 \quad g_1 \quad g_2 \quad \cdots \quad g_u\ ]^T$;
$p^{(l)}, p^{(u)} := F^{(l)}.G, F^{(u)}.G$;

[4:Combining the terms]
$q := \mathrm{Quo}(X^{z-2} p^{(u)}, s)$
return $\mathrm{Quo}(\mathrm{Rem}(X^{k-2} p^{(l)}, X^{k+1}), X^k) - \mathrm{Quo}(\mathrm{Rem}(q X^{k-2} s_l, X^{k+1}), X^k)$

Figure 2.5: Algorithm `PartialProduct`(f, g, s, d, k)

Equation (2.16) summarizes Algorithm `PartialProduct`(f, g, s, d, k) in Figure 2.5. This algorithm is almost same as Algorithm `PartialProduct`(f, g, s, d) in Figure 2.2 but for slight modifications.

Similarly, analogous to Algorithm `PartialProduct`(h, A, B, s, d) in Figure 2.4, we can describe Algorithm `PartialProduct`(h, A, B, s, d, k) in Figure 2.6, to compute

$$\mathrm{Quo}(\mathrm{Rem}(\mathrm{Rem}(hAB, s), X^{k+1}), X^k),$$

where $A \in \mathsf{K}[x]^{n\times\ell}$, $B \in \mathsf{K}[x]^{\ell\times m}$ and $h \in \mathsf{K}[x]$.

Using cost estimates similar to one used in lemma 2.8, we can state the following results

Lemma 2.10. *The cost of Algorithm* `PartialProduct`(h, A, B, s, d, k) *in Figure 2.6, is bounded by* $O(1/d\,(\deg A + \deg B)\mathsf{M}(\deg s) + nl\mathsf{M}(\deg A + \deg B) + \mathsf{MM}(n, \ell \deg B/d, m, d))$ *basic operations from* K. *Here* $A \in \mathsf{K}[x]^{n\times\ell}$ *and* $B \in \mathsf{K}[x]^{\ell\times m}$.

Corollary 2.11. *If* $A \in \mathsf{K}[x]^{n\times\ell}, B \in \mathsf{K}[x]^{\ell\times m}$, $\deg B \times \ell, \deg B \times \ell \in O(nd)$ *and* $\deg s \leq nd$, *then the cost of Algorithm* `PartialProduct`(h, A, B, s, d, k) *in Figure 2.6, is bounded by* $O(n\mathsf{M}(nd) + n^\omega\,\mathsf{M}(d))$ *basic operations from* K.

`PartialProduct`(h, A, B, s, d, k)
Input: $A \in K[x]^{n\times l}, B \in K[x]^{l\times m}, s, h \in K[x], k, d \in \mathbb{Z}_{>0}$
Condition: $s \neq 0$, $h = \mathrm{Rem}(h, s)$, $A = \mathrm{Rem}(A, s)$ $B = \mathrm{Rem}(B, s)$, $kd < \deg s$
Output: $\mathrm{Quo}(\mathrm{Rem}(\mathrm{Rem}(hAB, s), X^{k+1}), X^k) \in \mathsf{K}[x]^{n\times m}$

if $\deg(s) \leq 2d$
  return $\mathrm{Rem}(\mathrm{Rem}(hAB, s), x^d)$;
fi
$u, t, z, X := \lfloor \frac{\deg(B)}{d} \rfloor, max\{\lfloor \frac{\deg(A)}{d} \rfloor, u\}, \lfloor \frac{\deg(s)}{d} \rfloor, x^d$;

[1 : Preprocessing]
$\hat{H} = [\ h \quad hX \quad hX^2 \quad \cdots \quad hX^{t+u+1}\ ]$;
$H := \mathrm{Rem}(\hat{H}, s)$;
$H^{(l)}, H^{(u)} := \mathrm{Quo}(\mathrm{Rem}(H, X^{k+1}), X^k), \mathrm{Quo}(H, X^{z-1})$;
$s_l = \mathrm{Quo}(\mathrm{Rem}(s, X^{k+1}), X^k)$

[2: MiddleProduct]
for each $f_{ij} = A_{ij}$
$F_{ij}^{(l)} := \mathrm{MiddleProduct}(f_{ij}, H^{(l)}, u+t+1, u+1, d) \in K[x]^{1\times u+1}$;
$F_{ij}^{(u)} := \mathrm{MiddleProduct}(f_{ij}, H^{(u)}, u+t+1, u+1, d) \in K[x]^{1\times u+1}$
od

[3: Matrix multiplication]
for $i = 1$ *to* l
  Write $B_{ij} = g_{j,0} + g_{j,1}X + \cdots + g_{j,u}X^u$, for $1 \leq j \leq m$.

$$G_i = \begin{bmatrix} g_{1,0} & g_{2,0} & \cdots & g_{m,0} \\ g_{1,1} & g_{2,1} & \cdots & g_{n,1} \\ \vdots & \vdots & \cdots & \vdots \\ g_{1,u} & g_{2,u} & \cdots & g_{m,u} \end{bmatrix} \in \mathsf{K}[x]^{(u+1)\times m}$$

od

$$G = \left[\begin{array}{c} G_1 \\ \hline \\ \vdots \\ \hline G_{l-1} \\ \hline \\ G_l \end{array}\right] \in \mathsf{K}[x]^{(u+1)l\times m},$$

$$F_M^{(l)}, F_M^{(u)} = \begin{bmatrix} F_{11}^{(l)} & F_{12}^{(l)} & \cdots & F_{1(l-1)}^{(l)} & F_{1l}^{(l)} \\ F_{21}^{(l)} & F_{22}^{(l)} & \cdots & F_{2(l-1)}^{(l)} & F_{2l}^{(l)} \\ \vdots & \vdots & & \vdots & \vdots \\ F_{n1}^{(l)} & F_{n2}^{(l)} & \cdots & F_{n(l-1)}^{(l)} & F_{nl}^{(l)} \end{bmatrix}, \begin{bmatrix} F_{11}^{(u)} & F_{12}^{(u)} & \cdots & F_{1(l-1)}^{(u)} & F_{1l}^{(u)} \\ F_{21}^{(u)} & F_{22}^{(u)} & \cdots & F_{2(l-1)}^{(u)} & F_{2l}^{(u)} \\ \vdots & \vdots & & \vdots & \vdots \\ F_{n1}^{(u)} & F_{n2}^{(u)} & \cdots & F_{n(l-1)}^{(u)} & F_{nl}^{(u)} \end{bmatrix}$$

$P^{(l)}, P^{(u)} := F_M^{(l)}.G, F_M^{(u)}.G \in \mathsf{K}[x]^{n\times m}$;

[4: Combining the terms]
$Q := \mathrm{Quo}(X^{z-2}P^{(u)}, s)$
return $\mathrm{Quo}(\mathrm{Rem}(X^{k-2}P^{(l)}, X^{k+1}), X^k) - \mathrm{Quo}(\mathrm{Rem}(QX^{k-2}s_l, X^{k+1}), X^k)$;

Figure 2.6: Algorithm `PartialProduct`(h, A, B, s, d, k)

Chapter 3

Reduced Smith Transform

In this chapter R represents either $\mathsf{K}[x]$ or $\mathsf{K}[x]/\langle s\rangle$, for $s \neq 0 \in \mathsf{K}[x]$. We need a few definitions before we describe the central problem dealt with in this chapter. Recall from Chapter 1, Definition 1.3, the notion of Smith form that $S \in \mathsf{R}^{n\times n}$ is said to be in Smith form if S can be written as $\mathrm{diag}(s_1, s_2, \ldots, s_n) \in \mathsf{R}^{n\times n}$ with

- $s_i \in \mathcal{A}(\mathsf{R})$ for all $1 \leq i \leq n$, and
- $s_i \mid s_{i+1}$ for all $1 \leq i < n$.

We now define the notion of a Smith transform, a reduced Smith transform, a Smith decomposition and a reduced Smith decomposition as follows:

Definition 3.1. *A Smith transform of $A \in \mathsf{R}^{n\times n}$ is a tuple (U, V) such that*

- *$U, V \in \mathsf{R}^{n\times n}$ are unimodular, and*
- *$UAV = S \in \mathsf{R}^{n\times n}$, where S is the Smith form of A.*

We call the tuple $(\bar{U}, \bar{V})$ a reduced Smith transform if $\bar{V}[*, i] = \mathrm{Rem}(V[*, i], s_i)$ and $\bar{U}[i, *] = \mathrm{Rem}(U[i, *], s_i)$, for $1 \leq i \leq n$, for some Smith transform (U, V).

Definition 3.2. *A Smith decomposition of $B \in \mathsf{R}^{n\times n}$ is a tuple (U, S, V) such that*

- *$U, V \in \mathsf{R}^{n\times n}$ are unimodular, and*
- *$B = USV \in \mathsf{R}^{n\times n}$, where S is the Smith form of B.*

We call the tuple $(\bar{U}, S, \bar{V})$ an s-reduced Smith decomposition if

- $B = \mathrm{Rem}(\bar{U}S\bar{V}, s)$, *and*
- $\deg \bar{U}[*, i] < \deg s/s_i$ *and* $\deg \bar{V}[i, *] < \deg s/s_i$, *for* $1 \leq i \leq n$,

where s is a multiple of the largest Smith invariant of B. When $s = s_n$, the largest invariant factor of B, we call the tuple $(\bar{U}, S, \bar{V})$ a reduced Smith decomposition of B.

Note that if $A \in \mathsf{K}[x]^{n\times n}$ with $\deg A \leq d$, then the size of a reduced Smith transform is *a priori* bounded by $O(n^2 d)$ elements from K. This is a direct consequence of the bounds on the degree of the invariant factors of A as given in Theorem 1.4.

Given a nonsingular $A \in \mathsf{K}[x]^{n\times n}$ along with $S = \mathrm{snf}(A) \in \mathsf{K}[x]^{n\times n}$, where $\deg A \leq d$, we want to find a reduced Smith transform of A. Our target cost is $O^{\sim}(n^\omega d)$ basic operations from K. It is not unreasonable to assume that we are given S, the Smith form of B, as there exists a Las Vegas algorithm that computes S, given B over $\mathsf{K}[x]$, in cost $O(n^\omega (\log n)^2 \mathsf{B}(d))$ [28]. We compute a reduced Smith transform of A by reducing the problem to the computation of a s_n-reduced Smith decomposition of $s_n A^{-1}$. The following lemma shows the connection between a reduced Smith transform of A and a s_n-reduced Smith decomposition of $s_n A^{-1}$.

Lemma 3.3. *Given a nonsingular $A \in \mathsf{K}[x]^{n\times n}$, and $S = \mathrm{snf}(A) \in \mathsf{K}[x]^{n\times n}$, with the condition that $\deg s_n A^{-1} < \deg s_n$. The following statements are equivalent:*

- *(U, V) is a reduced Smith transform of A,*
- *$(VP, Ps_nS^{-1}P, PU)$ is a s_n-reduced Smith decomposition of s_nA^{-1},*

where $P \in \mathsf{K}[x]^{n\times n}$ is a permutation matrix with 1 on the antidiagonal.

Proof. Let (W, Y) be a Smith transform of A. Then

$$\begin{aligned} & WAY && = S \in \mathsf{K}[x]^{n\times n} \\ \Leftrightarrow\ & s_nA^{-1} && = Ys_nS^{-1}W \in \mathsf{K}[x]^{n\times n} \\ \Leftrightarrow\ & s_nA^{-1} && = YP(Ps_nS^{-1}P)PW \in \mathsf{K}[x]^{n\times n}. \end{aligned} \tag{3.1}$$

Note that

$$Ps_nS^{-1}P = \mathrm{snf}(s_nA^{-1}) = \mathrm{diag}(s_n/s_n, s_n/s_{n-1}, \ldots, s_n/s_1). \tag{3.2}$$

This implies that (YP, PW) is a Smith decomposition of s_nA^{-1}. Therefore, (W, Y) is a Smith transform of A if and only if (YP, PW) is a Smith decomposition of s_nA^{-1}. Let w_i be the i-th row of W and y_i be the i-th column of Y, for $1 \leq i \leq n$. Equation (3.1) can now be written as

$$s_nA^{-1} = \frac{s_n}{s_n} y_n w_n + \frac{s_n}{s_{n-1}} y_{n-1} w_{n-1} + \cdots + \frac{s_n}{s_1} y_1 w_1. \tag{3.3}$$

Consider taking equation (3.3) modulo s_n. The left hand side remains unchanged as we assume that $\deg s_n A^{-1} < \deg s_n$. Since each $y_i w_i$ is scaled by s_n/s_i, the equation will still hold if entries in w_i and y_i are reduced modulo s_i to $\mathrm{Rem}(w_i, s_i)$ and $\mathrm{Rem}(y_i, s_i)$, $1 \leq i \leq n$. Let $U = [w_1^T \mid w_2^T \cdots \mid w_n^T]^T$ and $V = [y_1 \mid y_2 \cdots \mid y_n]$. A s_n reduced Smith decomposition of $s_n A^{-1}$ is

$$s_n A^{-1} = \mathrm{Rem}(VP(Ps_nS^{-1}P)PU, s_n),$$

with $\deg((VP)[*, i]) < \deg s_{n-i+1}$ and $\deg((PU)[i, *]) < \deg s_{n-i+1}$ for $1 \leq i \leq n$, which simplifies to $\deg V[*, i] < \deg s_i$ and $\deg U[i, *] < \deg s_i$, $1 \leq i \leq n$. This is equivalent to stating that a reduced Smith transformation of A is (U, V) where $U[i, *] = \mathrm{Rem}(W[i, *], s_i)$ and $V[*, i] = \mathrm{Rem}(Y[*, i], s_i)$. □

Note that if A is row reduced, then $\deg s_n A^{-1} < \deg s_n$.

We can infer from the above lemma that, for a nonsingular $A \in \mathsf{K}[x]^{n \times n}$ with $\deg A = d$, the *a priori* size of a s_n-reduced Smith decomposition of $s_n A^{-1}$ is same as that of a reduced Smith transform of A and is bounded by $O(n^2 d)$ elements from K. This is a direct consequence of the bounds on the degree of each invariant factor of A (Theorem 1.4).

From now on we focus on computing a reduced Smith decomposition of $s_n A^{-1}$ modulo s_n. A iterative algorithm for computing a reduced Smith decomposition of $s_n A^{-1}$ is given in [30]. It computes a s_n-reduced Smith decomposition of $s_n A^{-1}$ in cost $O(n^2\, \mathsf{B}(nd))$ operations from K, when $\deg s_n A^{-1} < \deg s_n$. We adapt the iterative algorithm in [30] to give an algorithm to compute a s_n-reduced Smith decomposition of $s_n A^{-1}$ in cost $O(n^\omega (\log n)^2\, \mathsf{B}(d))$.

In the Section 3.1, we discuss the problem of computing a reduced Smith decomposition of nonsingular $B \in \mathsf{K}[x]^{n \times n}$, where $S = \mathrm{diag}(s_1, s_2, \cdots, s_n)$ is the Smith form of B. In Section 3.2 we adapt the techniques developed in Section 3.1 to compute a s_n-reduced Smith transform of $s_n A^{-1}$ from A and S, where $\deg A \leq d$ and $\mathrm{snf}(A) = S = \mathrm{diag}(s_1, s_2, \ldots, s_n)$. Our target cost is $O\tilde{}(n^\omega d)$ basic operations from K. We cannot achieve the target cost by directly working over $s_n A^{-1}$ as the size of $s_n A^{-1}$ in terms of number of elements from K can potentially be $O(n^3 d)$. To achieve the target cost, algorithm in Section 3.2 takes A as an input and works implicitly on $s_n A^{-1}$ by using the high order lifting technique [28] and the partial product technique from Chapter 2.

3.1 Reduced Smith Decomposition of B

Given a nonsingular $B \in \mathsf{K}[x]^{n \times n}$ and $S = \mathrm{diag}(s_1, s_2, \ldots, s_n) \in \mathsf{K}[x]^{n \times n}$, the Smith form of B, we want to compute (U, V) a s_n-reduced Smith decomposition of B :

$$B = USV \mod s_n. \tag{3.4}$$

Let $\mathsf{R} = \mathsf{K}[x]/\langle s_n \rangle$ (refer to Section 1.2 for details). Recall from Section 1.2, ϕ_{s_n} is the homomorphism from $\mathsf{K}[x]$ to R defined by $\phi_{s_n}(b) = \text{Rem}(b, s_n) \in \mathsf{R}$, for $b \in \mathsf{K}[x]$. Let $\bar{B} = \phi_{s_n}(B) \in \mathsf{R}^{n \times n}$. A reduced Smith decomposition (U, V) of $B \in \mathsf{K}[x]^{n \times n}$ satisfying equation (3.4) can be obtained by computing a Smith decomposition of $\bar{B}$ over the ring R. From now on we concern ourselves with computing a Smith decomposition of $\bar{B}$ over the ring R. We make the following assumption about $\bar{B} \in \mathsf{R}^{n \times n}$:

$$\text{snf}(\bar{B}_{[1,\ldots,i|1,\ldots,i]}) = \text{diag}(s_1, s_2, \cdots, s_i) \in \mathsf{R}^{n \times n} \tag{3.5}$$

for all $1 \leq i \leq n$. For a sufficiently large field K, any $B \in \mathsf{K}[x]^{n \times n}$, and unit upper and unit lower triangular matrices $U, L \in \mathsf{K}[x]^{n \times n}$ where the nonzero, non diagonal entries are chosen at random from K, $\phi_{s_n}(UBL)$ satisfies the equation (3.5) with high probability [20]. Thus the assumption (3.5) is not a limiting assumption.

Subsection 3.1.1 presents an iterative method to compute a Smith decomposition of $\bar{B}$ in $O(n^3)$ operations from R. Subsection 3.1.2 builds on Subsection 3.1.1 to compute a Smith decomposition of $\bar{B}$ in $O(n^\omega)$ operations from R.

3.1.1 Iterative Approach

Let $\mathsf{R} = \mathsf{K}[x]/\langle s \rangle$, $0 \neq s \in \mathsf{K}[x]$. Given $B \in \mathsf{R}^{n \times n}$ and $S = \text{snf}(B) \in \mathsf{R}^{n \times n}$, we outline a algorithm for finding a Smith decomposition of B, where B satisfies property (3.5). We also assume that none of the invariant factors of B are zero. This is not a limiting assumption; if only the first r invariant factors of B are nonzero, then using assumption (3.5) we can work over the smaller matrix $B_{[1,\ldots,r|1,\ldots,r]}$ to get a Smith decomposition of B.

Let $B^{(1)} = B \in \mathsf{R}^{n \times n}$. Due to (3.5), there is a unit $t_1 \in \mathsf{R}$ such that

$$t_1 \times B^{(1)}[1,1] = s_1 \in \mathsf{R}^{n \times n}.$$

Let

$$C^{(1)} = \text{diag}(t_1, 1, 1, \ldots, 1) B^{(1)}.$$

Note that

$$C^{(1)} \equiv_L B^{(1)} \in \mathsf{R}^{n \times n}. \tag{3.6}$$

By Theorem 1.4, s_1 divides all the entries of $C^{(1)}$, hence using elementary row and column operations we can eliminate the non diagonal entries in the first row and first column of $C^{(1)}$ to get $B^{(2)}$:

$$\left[\begin{array}{c|c} 1 & 0 \\ \hline -C^{(1)}_{[2,\ldots,n|1]}/s_1 & I \end{array}\right] C^{(1)} \left[\begin{array}{c|c} 1 & -C^{(1)}_{[1|2,\ldots,n]}/s_1 \\ \hline 0 & B^{(2)} \end{array}\right] = \left[\begin{array}{c|c} s_1 & 0 \\ \hline 0 & B^{(2)} \end{array}\right] \in \mathsf{R}^{n \times n}.$$

We are careful of the fact that for $a, b \in \mathsf{R}$, the operation a/b is not defined for a nonunit b with any $a \in \mathsf{R}$. We only use a/b when ever we are sure that b divides a in R and $b \in \mathcal{A}(\mathsf{R})$. Using (3.6) we get

$$\left[\begin{array}{c|c} t_1 & 0 \\ \hline -C^{(1)}_{[2,\ldots,n|1]}/s_1 & I \end{array}\right] B^{(1)} \left[\begin{array}{c|c} 1 & -C^{(1)}_{[1|2,\ldots,n]}/s_1 \\ \hline 0 & B^{(2)} \end{array}\right] = \left[\begin{array}{c|c} s_1 & 0 \\ \hline 0 & B^{(2)} \end{array}\right] \in \mathsf{R}^{n\times n}.$$

Thus $B^{(1)}$ decomposes in the following manner:

$$B^{(1)} = \overbrace{\left[\begin{array}{c|c} t_1^{-1} & 0 \\ \hline t_1^{-1}C^{(1)}_{[2,\ldots,n|1]}/s_1 & I \end{array}\right]}^{U^{(1)}} \left[\begin{array}{c|c} s_1 & 0 \\ \hline 0 & B^{(2)} \end{array}\right] \overbrace{\left[\begin{array}{c|c} 1 & t_1^{-1}C^{(1)}_{[1|2,\ldots,n]}/s_1 \\ \hline 0 & B^{(2)} \end{array}\right]}^{V^{(1)}} \in \mathsf{R}^{n\times n}.$$

We know from (3.5)

$$\mathrm{snf}(B^{(1)}_{[1,2|1,2]}) = \mathrm{diag}(s_1, s_2).$$

Therefore $B^{(2)}[1,1]$ must be an associate of s_2. Thus we follow the same procedure on $B^{(2)}$ and so on to get a complete Smith decomposition.

For $B \in \mathsf{R}^{3\times 3}$, the matrix B decomposes in the following manner:

$$\overbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}^{B^{(1)}} = \overbrace{\begin{bmatrix} t_1^{-1} & & \\ * & 1 & \\ * & & 1 \end{bmatrix}}^{U^{(1)}} \overbrace{\begin{bmatrix} s_1 & & \\ & * & * \\ & * & * \end{bmatrix}}^{\mathrm{diag}(s_1, B^{(2)})} \overbrace{\begin{bmatrix} 1 & * & * \\ & 1 & \\ & & 1 \end{bmatrix}}^{V^{(1)}} = \overbrace{\begin{bmatrix} 1 & & \\ & t_2^{-1} & \\ & * & 1 \end{bmatrix}}^{U^{(2)}} \overbrace{\begin{bmatrix} t_1^{-1} & & \\ * & 1 & \\ * & & 1 \end{bmatrix}}^{U^{(1)}} \overbrace{\begin{bmatrix} s_1 & & \\ & s_2 & \\ & & * \end{bmatrix}}^{\mathrm{diag}(s_1, s_2, B^{(3)})} \overbrace{\begin{bmatrix} 1 & * & * \\ & 1 & \\ & & 1 \end{bmatrix}}^{V^{(1)}} \overbrace{\begin{bmatrix} 1 & & \\ & 1 & * \\ & & 1 \end{bmatrix}}^{V^{(2)}}$$

$$= \overbrace{\begin{bmatrix} 1 & & \\ & 1 & \\ & & t_3^{-1} \end{bmatrix}}^{U^{(3)}} \overbrace{\begin{bmatrix} 1 & & \\ & t_2^{-1} & \\ & * & 1 \end{bmatrix}}^{U^{(2)}} \overbrace{\begin{bmatrix} t_1^{-1} & & \\ * & 1 & \\ * & & 1 \end{bmatrix}}^{U^{(1)}} \overbrace{\begin{bmatrix} s_1 & & \\ & s_2 & \\ & & s_3 \end{bmatrix}}^{\mathrm{diag}(s_1, s_2, s_3)} \overbrace{\begin{bmatrix} 1 & * & * \\ & 1 & \\ & & 1 \end{bmatrix}}^{V^{(1)}} \overbrace{\begin{bmatrix} 1 & & \\ & 1 & * \\ & & 1 \end{bmatrix}}^{V^{(2)}} \overbrace{\begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}}^{V^{(3)}}$$

Finally we get the Smith Decomposition as:

$$= \overbrace{\begin{bmatrix} t_1^{-1} & & \\ * & t_2^{-1} & \\ * & * & t_3^{-1} \end{bmatrix}}^{U} \overbrace{\begin{bmatrix} s_1 & & \\ & s_2 & \\ & & s_3 \end{bmatrix}}^{\mathrm{diag}(s_1, s_2, s_3)} \overbrace{\begin{bmatrix} 1 & * & * \\ & 1 & * \\ & & 1 \end{bmatrix}}^{V},$$

where $U = U^{(3)}U^{(2)}U^{(1)}$ and $V = V^{(1)}V^{(2)}V^{(3)}$. We obtain the following result:

Lemma 3.4. *Let* $\mathsf{R} = \mathsf{K}[x]/\langle s\rangle$, $s \neq 0 \in \mathsf{K}[x]$. *If* $B \in \mathsf{R}^{n\times n}$, *with* $S = \mathrm{snf}(B) \in \mathsf{R}^{n\times n}$, *satisfies the condition that:*

$$\mathrm{snf}(B_{[1,\ldots,i|1,\ldots,i]}) = \mathrm{diag}(s_1, s_2, \cdots, s_i) \in \mathsf{R}^{n\times n}$$

for all $1 \leq i \leq n$*, then* B *admits a Smith decomposition* (U, S, V) *over* R*, where* U *is unimodular lower triangular and* V *is unit upper triangular. We can compute this Smith decomposition in* $O(n^3)$ *basic operations from* R*.*

In the next section, we introduce matrix multiplication and try to reduce the number of operations over R to $O(n^\omega)$.

3.1.2 Recursive Approach

As in Subsection 3.1.1, let $\mathsf{R} = \mathsf{K}[x]/\langle s \rangle$, $0 \neq s \in \mathsf{K}[x]$. Given $B \in \mathsf{R}^{n \times n}$ and $S = \mathrm{snf}(B) \in \mathsf{R}^{n \times n}$, we outline a recursive algorithm for finding a Smith decomposition of B, where B satisfies property (3.5). We assume that none of the invariant factors of B are zero. This is not a limiting assumption; if only the first r invariant factors of B are nonzero, then using assumption (3.5) we can work over the smaller matrix $B_{[1,\ldots,r|1,\ldots,r]}$ to get a Smith decomposition. The algorithm returns "fail" if (3.5) is not satisfied. Let this algorithm be called $\mathtt{SD}(B, S, n, s)$. Algorithm $\mathtt{SD}$ works over R. Let us write B in the following manner:

$$B = \left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array}\right] \in \mathsf{R}^{n \times n},$$

where $B_1 \in \mathsf{R}^{n_1 \times n_1}$, for some $0 < n_1 < n$. Applying recursively $\mathtt{SD}$ on B_1 we get:

$$U_1, S_1, V_1 = \mathtt{SD}(B_1, S_{[1,\ldots,n_1|1,\ldots,n_1]}, n_1, s) \in \mathsf{R}^{n_1 \times n_1}.$$

We then apply the following unimodular transformation on $\bar{B}$:

$$\left[\begin{array}{c|c} U_1^{-1} & \\ \hline & I \end{array}\right] \left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array}\right] \left[\begin{array}{c|c} V_1^{-1} & \\ \hline & I \end{array}\right] = \left[\begin{array}{c|c} S_1 & \hat{B}_2 \\ \hline \hat{B}_3 & B_4 \end{array}\right] \in \mathsf{R}^{n \times n}.$$

As in Subsection 3.1.1, due to Theorem 1.4 every element in $\hat{B}_2[i, *]$ and $\hat{B}_3[*, i]$ is divisible by $S_1[i, i]$, $1 \leq i \leq n_1$. This suggests that $S_1^{-1}\hat{B}_2 \in \mathsf{R}^{n_1 \times (n-n_1)}$ and $\hat{B}_3 S_1^{-1} \in \mathsf{R}^{(n-n_1) \times n_1}$. Thus we can apply the following unimodular operation:

$$\left[\begin{array}{c|c} I & \\ \hline -\hat{B}_3 S_1^{-1} & I \end{array}\right] \left[\begin{array}{c|c} S_1 & \hat{B}_2 \\ \hline \hat{B}_3 & B_4 \end{array}\right] \left[\begin{array}{c|c} I & -S_1^{-1}\hat{B}_2 \\ \hline & I \end{array}\right] = \left[\begin{array}{c|c} S_1 & \\ \hline & \hat{B}_4 \end{array}\right] \in \mathsf{R}^{n \times n}.$$

Therefore we reduce the matrix B in the following manner:

$$\left[\begin{array}{c|c} U_1^{-1} & \\ \hline -B_3 V_1^{-1} S_1^{-1} & I \end{array}\right] \left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array}\right] \left[\begin{array}{c|c} V_1^{-1} & -S_1^{-1}U_1^{-1}B_2 \\ \hline & I \end{array}\right] = \left[\begin{array}{c|c} S_1 & \\ \hline & \hat{B}_4 \end{array}\right] \in \mathsf{R}^{n \times n}.$$

$\mathtt{SD}(B, S, n, s)$
$\mathsf{R} = \mathsf{K}[x]/\langle s \rangle$.
Input: $B \in \mathsf{R}^{n\times n}$, $S = \mathrm{snf}(B) \in \mathsf{R}^{n\times n}$ with $S[n,n] \neq 0$, $n \in \mathbb{Z}_{>0}, 0 \neq s \in \mathsf{K}[x]$.
Condition: : $\mathrm{snf}(B_{[1,\ldots,i|1,\ldots,i]}) = \mathrm{diag}(s_1, s_2, \cdots, s_i) \in R^{i\times i}$, $1 \leq i \leq n$.
Output: $U, S, V \in \mathsf{R}^{n\times n} : A = USV \in \mathsf{R}^{n\times n}$ or fail if the condition above is not satisfied.

if $n = 1$ **then**

$t_1 := \mathrm{Unit}(B[1,1])$;
if $t_1 B[1,1] \neq S[1,1]$ **then return** fail **fi**
return (t_1, S, I)
fi
$n_1 := \lfloor n/2 \rfloor$;
$U_1, S_1, V_1 := \mathtt{SD}(B_{[1,\ldots,n_1|1,\ldots,n_1]}, S_{[1,\ldots,n_1|1,\ldots,n_1]}, n_1, s)$;

$$\left[\begin{array}{c|c} S_1 & \\ \hline & \hat{B}_4 \end{array}\right] = \left[\begin{array}{c|c} U_1^{-1} & \\ \hline -B_3 V_1^{-1} S_1^{-1} & I \end{array}\right] B \left[\begin{array}{c|c} V_1^{-1} & -S_1^{-1}U_1^{-1}B_2 \\ \hline & I \end{array}\right] \in R^{n\times n};$$

$U_2, S_2, V_2 := \mathtt{SD}\Big(\hat{B}_4, S_{[n_1+1,\ldots,n|n_1+1,\ldots,n]}, n-n_1, s\Big)$;

return $\left[\begin{array}{c|c} U_1 & \\ \hline B_3 V_1^{-1} S_1^{-1} U_1 & U_2 \end{array}\right], S, \left[\begin{array}{c|c} V_1 & V_1 S_1^{-1} U_1^{-1} B_2 \\ \hline & V_2 \end{array}\right] \in \mathsf{R}^{n\times n}$;

Figure 3.1: `Smith Decomposition`

This also gives the following decomposition of B:

$$\left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array}\right] = \left[\begin{array}{c|c} U_1 & \\ \hline B_3 V_1^{-1} S_1^{-1} U_1 & I \end{array}\right] \left[\begin{array}{c|c} S_1 & \\ \hline & \hat{B}_4 \end{array}\right] \left[\begin{array}{c|c} V_1 & V_1 S_1^{-1} U_1^{-1} B_2 \\ \hline & I \end{array}\right] \in \mathsf{R}^{n\times n}.$$

Now we compute a Smith decomposition of $\hat{B}_4$:

$$U_2, S_2, V_2 = \mathtt{SD}(\hat{B}_4, S_{[n_1+1,\ldots,n|n_1+1,\ldots,n]}, n-n_1, s).$$

Finally we compile the result and give a Smith decomposition of B as:

$$B = \left[\begin{array}{c|c} U_1 & \\ \hline B_3 V_1^{-1} S_1^{-1} U_1 & U_2 \end{array}\right] S \left[\begin{array}{c|c} V_1 & V_1 S_1^{-1} U_1^{-1} B_2 \\ \hline & V_2 \end{array}\right] \in \mathsf{R}^{n\times n}.$$

$\mathtt{SD}(B, S, 1, s)$ is the base case, where $B, S \in \mathsf{R}^{1\times 1}$. If there exists a unit t such that $tB[1,1] = S[1,1]$, we return $(t, S[1,1], I)$ else we return fail. Figure 3.1 summarizes Algorithm `SD`.

We now analyze the cost of Algorithm $\mathtt{SD}(B, S, n, s)$ in Figure 3.1. The multiplication of B_3 with V_1^{-1} and $B_3 V_1^{-1} S_1^{-1}$ with U_1 will cost $\mathsf{MM}(n-n_1, n_1, n_1)$ operations from R

each. Similarly the multiplication of U_1^{-1} with B_2 and $V_1 S_1^{-1} U_1^{-1}$ with B_2 can be computed in $\mathsf{MM}(n_1, n-n_1, n_1)$ operations from R each. We need $n_1(n-n_1)$ operations from R to apply S^{-1}. Thus in all we need $2\mathsf{MM}(n-n_1, n_1, n_1) + 2\mathsf{MM}(n_1, n-n_1, n_1) + 2n_1(n-n_1)$ operations from R in each recursive call to $\mathtt{SD}(B, S, n, s)$. If $n_1 = \lfloor n/2 \rfloor$ and $T(n)$ is the bound on the cost of $\mathtt{SD}(B, S, n, s)$ in terms of the number of basic operations from R, we get the following relation:

$$T(n) \leq T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + cn^{\omega},$$

for some constant $c \in \mathbb{Z}_{>0}$ and $T(1) = 1$. This implies that $T(n)$ is bounded by $O(n^{\omega})$. We obtain the following result

Lemma 3.5. *Let $\mathsf{R} = \mathsf{K}[x]/\langle s \rangle$, $0 \neq s \in \mathsf{K}[x]$. Given $B \in \mathsf{R}^{n \times n}$ satisfying (3.5) and $S \in \mathsf{R}^{n \times n}$, the Smith normal form of B, with $S[n,n] \neq 0$. Algorithm* SD *in Figure 3.1, returns a Smith decomposition (U, S, V) of B in $O(n^{\omega})$ basic operations from $\mathsf{K}[x]/\langle s \rangle$.* SD *returns fail if S is not the Smith normal form of B.*

3.2 Reduced Smith Decomposition of $s_n A^{-1}$

In Section 3.1, we discussed an algorithm to find a reduced Smith decomposition of $B \in \mathsf{K}[x]^{n \times n}$, given nonsingular B and S, the Smith form of S, by finding a Smith decomposition of $\phi_{s_n}(B)$ over the ring $\mathsf{R} = \mathsf{K}[x]/\langle s_n \rangle$. Let nonsingular $A \in \mathsf{K}[x]^{n \times n}$ with $\deg A \leq d$ and $S = \mathrm{snf}(A) = \mathrm{diag}(s_1, s_2, \ldots, s_n) \in \mathsf{K}[x]^{n \times n}$. In this section we adapt the technique used in Section 3.1 to compute a s_n-reduced Smith decomposition of $s_n A^{-1}$, given A and S, i.e., we want to compute $U, V \in \mathsf{K}[x]$ such that:

$$s_n A^{-1} \equiv U \mathrm{snf}(s_n A^{-1}) V \mod s_n.$$

If we directly apply Algorithm SD in Figure 3.1 on $s_n A^{-1}$, the cost of computing a s_n-reduced Smith decomposition will be $O(n^{\omega}\, \mathsf{B}(nd))$. This estimate uses the bound that $\deg s_n \leq nd$ (Theorem 1.4) and the fact stated in Section 1.1.1, that every basic operation in R can be done in $O(\deg s_n)$ operations from K. In this Chapter, we reduce the cost of computing a s_n-reduced Smith decomposition of $s_n A^{-1}$ to $O\tilde{}(n^{\omega} d)$ basic operations from K. We know that

$$\mathrm{snf}(s_n A^{-1}) = \mathrm{diag}(s_n/s_n, s_n/s_{n-2}, \ldots, s_n/s_1) \in \mathsf{K}[x]^{n \times n}.$$

We make the fast multiplication assumption (1.6) in this section: $\mathsf{M}(t) \in O(t^{\omega-1})$. Also, in line with Lemma 3.3 we assume that

$$\deg s_n A^{-1} < \deg s_n. \tag{3.7}$$

As in Section 3.1, we work over the ring $\mathsf{R} = \mathsf{K}[x]/\langle s_n \rangle$ to find a Smith Decomposition of $\phi_{s_n}^{-1}(s_n A^{-1})$ which translates into a s_n-reduced Smith decomposition of $s_n A^{-1}$ over $\mathsf{K}[x]$. Let $\overline{s_n A^{-1}} = \phi_{s_n}^{-1}(s_n A^{-1})$. Analogous to (3.5), we make the assumption that

$$\mathrm{snf}(\overline{s_n A^{-1}}_{[1,\ldots,i|1,\ldots,i]}) = \mathrm{diag}\left(\frac{s_n}{s_n}, \frac{s_n}{s_{n-1}}, \ldots, \frac{s_n}{s_{n-i+1}}\right) \in \mathsf{R}^{i\times i}, \tag{3.8}$$

for $1 \leq i \leq n$. As stated in Section 3.1 this is not a limiting assumption. For a sufficiently large field K, any $s_n A^{-1} \in \mathsf{K}[x]^{n\times n}$, and unit upper and unit lower triangular matrices $U, L \in \mathsf{K}[x]^{n\times n}$ where the nonzero, non diagonal entries are chosen at random from K, $\phi_{s_n}(U s_n A^{-1} L)$ satisfies the equation (3.8) with high probability [20]. Applying U, L to $s_n A^{-1}$ to get $U s_n A^{-1} L$ is same as applying L^{-1}, U^{-1} to A and get $\hat{A} = L^{-1} A U^{-1}$ and then computing $s_n \hat{A}^{-1}$, i.e., $U s_n A^{-1} L = s_n \hat{A}^{-1}$.

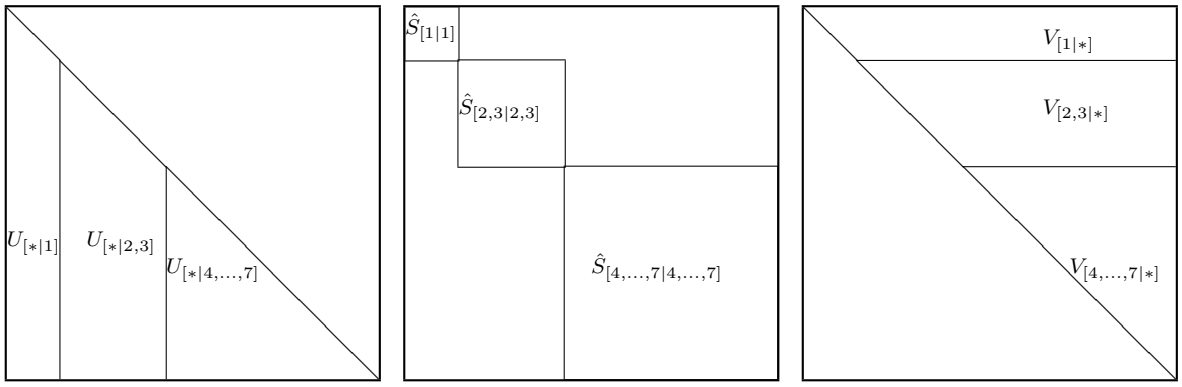
In Subsection 3.2.1, we work on $s_n A^{-1}$ directly and introduce ideas which exploit the structure of $s_n A^{-1}$. In Subsection 3.2.2 we introduce ideas to work without having $s_n A^{-1}$ explicitly, rather we will compute the parts of $s_n A^{-1}$ needed in each step and in this manner achieve the target cost bound of $O^{\tilde{}}(n^\omega d)$ operations from K.

3.2.1 Blocking

We are given a matrix $s_n A^{-1} \in \mathsf{K}[x]^{n\times n}$ satisfying (3.8) and (3.7), and $\hat{S} \in \mathsf{K}[x]^{n\times n}$, the Smith normal form of $s_n A^{-1}$. We develop an algorithm to find a s_n-reduced Smith decomposition of $s_n A^{-1}$ by computing a Smith decomposition $(U, \phi_{s_n}(\hat{S}), V)$ of $\overline{s_n A^{-1}}$ over R. The algorithm returns "fail" if (3.8) is not satisfied. The algorithm works over R. For the sake of simplicity, let us assume that

$$n = \sum_{i=0}^{t} 2^i = 2^{t+1} - 1. \tag{3.9}$$

This assumption can be made without loss of generality as we can always augment a matrix $s_n A^{-1}$ to $\mathrm{diag}(I, s_n A^{-1})$ to fulfill (3.9), where the dimension of I is less than the dimension of $s_n A^{-1}$. We know from Lemma 3.4 that $\overline{s_n A^{-1}}$ has a Smith decomposition $(U, \phi_{s_n}(\hat{S}), V)$, where U is lower triangular and V is upper triangular. We divide the columns of $\overline{s_n A^{-1}}$ and U into blocks of sizes $1, 2, 2^2, \ldots, 2^t$. Similarly we divide the rows of V into blocks of sizes $1, 2, 2^2, \ldots, 2^t$. The following diagram shows the division of $U, \hat{S}, V \in \mathsf{R}^{7\times 7}$.



As explained at the beginning of this chapter, the column $U_{[*|i]}$ is bounded by $\deg s_i$, which by Theorem 1.4 is bounded by $nd/(n-i+1)$. Thus, the *a priori* bounds on the size of each block is $O(n^2d)$ symbols from K. If we denote by precision the *a priori* bound on the degree, then the dimension of each block times the precision will be bounded by $O(n^2d)$. The idea to partition a problem into $\log n$ subproblems of size $O(n^2d)$, and compute each subproblem in $O^{\sim}(n^\omega d)$ basic operations from K to solve the whole problem in cost $O^{\sim}(n^\omega d)$ operations from K is also used in [12]. In Subsection 3.1.1 we recovered j-th Smith invariant of B in the j-th iteration. Now, we recover $(s_n/s_{n-2^j+1}, s_n/s_{n-2^j}, \ldots, s_n/s_{n-2^{j+1}+2})$ and corresponding columns and rows of U and V in the j-th iteration, j starting from 0 and going up to t.

We start with $B^{(0)} = \overline{s_n A^{-1}}$. At iteration $t = 0$ we recover s_n/s_n in exactly the same manner as done in Subsection 3.1.1. At the end of iteration $t = 0$, we have unimodular U, V and $B^{(1)}$:

$$\overline{s_n A^{-1}} = U \left[\begin{array}{c|c} s_n/s_n & \\ \hline & B^{(1)} \end{array}\right] V \in \mathsf{R}^{n\times n}. \tag{3.10}$$

For the next iteration, we try to recover the next two invariant factors. From (3.10) and (3.5), we know that:

$$\begin{aligned} \mathrm{snf}(B^{(1)}_{[1|1]}) &= s_n/s_{n-1}, \\ \mathrm{snf}(B^{(1)}_{[1,2|1,2]}) &= \mathrm{diag}(s_n/s_{n-1}, s_n/s_{n-2}). \end{aligned}$$

Using Algorithm 3.1, `SD`, we can find unimodular $T_l^{(1)}, T_r^{(1)} \in \mathsf{R}^{2\times 2}$ such that

$$B^{(1)}_{[1,2|1,2]} = T_l^{(1)} \mathrm{diag}(s_n/s_{n-1}, s_n/s_{n-2}) T_r^{(1)}.$$

Let

$$C^{(1)} = \left[\begin{array}{c|c} (T_l^{(1)})^{-1} & \\ \hline & I_{n-1} \end{array}\right] B^{(1)} \left[\begin{array}{c|c} (T_r^{(1)})^{-1} & \\ \hline & I_{n-3} \end{array}\right].$$

Thus

$$C^{(1)}_{[1,2|1,2]} = \mathrm{diag}(s_n/s_{n-1}, s_n/s_{n-2}).$$

As is clear from the iterative approach in Subsection 3.1.1, s_n/s_{n-1} divides all the entries in the first row and first column of $C^{(1)}$ and s_n/s_{n-2} divides all the entries in the second row and second column of $C^{(1)}$. We can infer that:

$$(C^{(1)}_{[1,2|1,2]})^{-1} C^{(1)}_{[1,2|*]} \in \mathsf{R}^{2\times(n-1)},$$

$$C^{(1)}_{[*|1,2]} (C^{(1)}_{[1,2|1,2]})^{-1} \in \mathsf{R}^{(n-1)\times 2}.$$

Thus we can eliminate the non diagonal entries in the first and second row and column of $C^{(1)}$ to get $B^{(2)}$:

$$\left[\begin{array}{cc|c} s_n/s_{n-1} & & \\ & s_n/s_{n-2} & \\ \hline & & B^{(2)} \end{array}\right] = \left[\begin{array}{cc|c} I_2 & & \\ \hline C^{(1)}_{[3,\ldots,n-1|1,2]}(C^{(1)}_{[1,2|1,2]})^{-1} & & I_{n-3} \end{array}\right] C^{(1)} \left[\begin{array}{cc|c} I_2 & & (C^{(1)}_{[1,2|1,2]})^{-1} C^{(1)}_{[1,2|3,\ldots,n-1]} \\ \hline & & I_{n-3} \end{array}\right].$$

Therefore $B^{(2)}$ can be recovered from $B^{(1)}$ in the following manner:

$$\left[\begin{array}{cc|c} s_n/s_{n-1} & & \\ & s_n/s_{n-2} & \\ \hline & & B^{(2)} \end{array}\right] = \left[\begin{array}{cc|c} (T^{(1)}_l)^{-1} & & \\ \hline C^{(1)}_{[3,\ldots,n-1|1,2]}(C^{(1)}_{[1,2|1,2]})^{-1} & & I_{n-3} \end{array}\right] B^{(1)} \left[\begin{array}{cc|c} (T^{(1)}_r)^{-1} & & (C^{(1)}_{[1,2|1,2]})^{-1} C^{(1)}_{[1,2|3,\ldots,n-1]} \\ \hline & & I_{n-3} \end{array}\right].$$

Thus, $B^{(1)}$ has been decomposed in the following fashion:

$$\overbrace{\left[\begin{array}{cc|c} T^{(1)}_l & & \\ & & \\ \hline -C^{(1)}_{[3,\ldots,n-1|1,2]}(C^{(1)}_{[1,2|1,2]})^{-1} T^{(1)}_l & & I_{n-3} \end{array}\right]}^{U^{(1)}} \left[\begin{array}{cc|c} s_n/s_{n-1} & & \\ & s_n/s_{n-2} & \\ \hline & & B^{(2)} \end{array}\right] \overbrace{\left[\begin{array}{cc|c} T^{(1)}_r & -T^{(1)}_r (C^{(1)}_{[1,2|1,2]})^{-1} C^{(1)}_{[1,2|3,\ldots,n-1]} & \\ \hline & & I_{n-3} \end{array}\right]}^{V^{(1)}},$$

where $\deg U^{(1)}, \deg V^{(1)} < \deg s_{n-1}$. We follow the same procedure on $B^{(2)}$ but with the aim to recover the next 4 entries and so on. Finally $\overline{s_n A^{-1}}$ will have the following Smith decomposition over R:

$$\overline{s_n A^{-1}} = U \hat{S} V,$$

where

$$U = \prod_{j=0}^{t} \left[\begin{array}{cc|c} I_{2^j-1} & & \\ \hline & & U^{(j)} \end{array}\right]$$

and

$$V = \prod_{j=t}^{0} \left[\begin{array}{cc|c} I_{2^j-1} & & \\ \hline & & V^{(j)} \end{array}\right].$$

For $s_n A^{-1} \in \mathsf{R}^{7\times 7}$, the shape of the decomposition is as follows:

$$I \overbrace{\begin{bmatrix} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \end{bmatrix}}^{s_7 A^{-1}} I = \begin{bmatrix} * & & & & & & \\ * & 1 & & & & & \\ * & & 1 & & & & \\ * & & & 1 & & & \\ * & & & & 1 & & \\ * & & & & & 1 & \\ * & & & & & & 1 \end{bmatrix} \begin{bmatrix} s_7/s_7 & & & & & & \\ & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & * & * & * & * & * & * \end{bmatrix} \begin{bmatrix} 1 & * & * & * & * & * & * \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} * & & & & & & \\ * & * & & & & & \\ * & * & * & & & & \\ * & * & * & 1 & & & \\ * & * & * & & 1 & & \\ * & * & * & & & 1 & \\ * & * & * & & & & 1 \end{bmatrix} \begin{bmatrix} s_n/s_7 & & & & & & \\ & s_7/s_6 & & & & & \\ & & s_7/s_5 & & & & \\ & & & * & * & * & * \\ & & & * & * & * & * \\ & & & * & * & * & * \\ & & & * & * & * & * \end{bmatrix} \begin{bmatrix} 1 & * & * & * & * & * & * \\ & 1 & * & * & * & * & * \\ & & 1 & * & * & * & * \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} * & & & & & & \\ * & * & & & & & \\ * & * & * & & & & \\ * & * & * & * & & & \\ * & * & * & * & * & & \\ * & * & * & * & * & * & \\ * & * & * & * & * & * & * \end{bmatrix} \begin{bmatrix} s_7/s_7 & & & & & & \\ & s_7/s_6 & & & & & \\ & & s_7/s_5 & & & & \\ & & & s_7/s_4 & & & \\ & & & & s_7/s_3 & & \\ & & & & & s_7/s_2 & \\ & & & & & & s_7/s_1 \end{bmatrix} \begin{bmatrix} 1 & * & * & * & * & * & * \\ & 1 & * & * & * & * & * \\ & & 1 & * & * & * & * \\ & & & 1 & * & * & * \\ & & & & 1 & * & * \\ & & & & & 1 & * \\ & & & & & & 1 \end{bmatrix}$$

The following procedure summarizes the whole process.

$B^{(0)}, U, V = \overline{s_n A^{-1}}, I_n, I_n \in \mathsf{R}^{n\times n}$
for j **from** 0 **to** t **do**

$\mathcal{J} = 1, 2, \ldots, 2^j$
$n_j = n - 2^j + 1$

[1. Find the Smith Decomposition]
$S^{(j)} = \mathrm{diag}(s_n/s_{n-2^j+1}, s_n/s_{n-2^j}, \ldots, s_n/s_{n-2^{j+1}}) \in \mathsf{R}^{2^j\times 2^j}$
$T_l^{(j)}, W, T_r^{(j)} = \mathtt{SD}(B^{(j)}_{[\mathcal{J}|\mathcal{J}]}, S^{(j)}, 2^j, s_{n_j}) \in \mathsf{R}^{2^j\times 2^j}$;

[2. Eliminate rows and columns]
$C^{(j)} := \mathrm{diag}((T_l^{(j)})^{-1}, I_{n-2^{j+1}+1}) B^{(j)} \mathrm{diag}((T_r^{(j)})^{-1}, I_{n-2^{j+1}+1}) \in \mathsf{R}^{n_j\times n_j}$;
$$\left[\begin{array}{c|c} S^{(j)} & \\ \hline & B^{(j+1)} \end{array}\right] := \left[\begin{array}{c|c} I_{2^j} & \\ \hline C^{(j)}_{[2^j+1,\ldots,n-1|\mathcal{J}]}(C^{(j)}_{[\mathcal{J}|\mathcal{J}]})^{-1} & I_{n-2^{j+1}+1} \end{array}\right] C^{(j)} \left[\begin{array}{c|c} I_{2^j} & (C^{(j)}_{[\mathcal{J}|\mathcal{J}]})^{-1} C^{(j)}_{[\mathcal{J}|2^j+1,\ldots,n-1]} \\ \hline & I_{n-2^j+1} \end{array}\right];$$

[3. Update]
$$U^{(j)} := \mathrm{Rem}\left(\left[\begin{array}{c|c} T_l^{(j)} & \\ \hline -C^{(j)}_{[2^j+1,\ldots,n-1|\mathcal{J}]}(C^{(j)}_{[\mathcal{J}|\mathcal{J}]})^{-1} T_l^{(j)} & I_{n-2^{j+1}+1} \end{array}\right], s_{n_j}\right) \in \mathsf{R}^{n_j\times n_j};$$

$V^{(j)} := \mathrm{Rem}\left(\left[\begin{array}{c|c} T_r^{(j)} & -T_r^{(j)}(C^{(j)}_{[\mathcal{J}|\mathcal{J}]})^{-1}C^{(j)}_{[\mathcal{J}|2^j+1,\ldots,n-1]} \\ \hline & I_{n-2^{j+1}+1} \end{array}\right], s_{n_j}\right) \in \mathsf{R}^{n_j\times n_j}$;

$U := U\left[\begin{array}{c|c} I_{2^j-1} & \\ \hline & U^{(j)} \end{array}\right] \in \mathsf{R}^{n\times n}$;

$V := \left[\begin{array}{c|c} I_{2^j-1} & \\ \hline & V^{(j)} \end{array}\right] V \in \mathsf{R}^{n\times n}$;

Assert: $\overline{s_nA^{-1}} = U\mathrm{diag}(S^{(0)},\ldots,S^{(j)},B^{(j+1)})V \in \mathsf{R}^{n\times n}$

od

In the j-th iteration, we can bound the cost of Phase 1 by $O(2^{j\omega})$ operations from R using Lemma 3.5. It translates to $O(2^{j\omega}\mathsf{B}(\deg s_n))$ basic operations from K. Using Theorem 1.4, the cost of Phase 1 simplifies to $O(2^{j\omega}\,\mathsf{B}(nd))$ basic operations from K. Phase 2 costs $O((n-2^j+1)^\omega)$ operations from R which can be achieved by $O((n-2^j+1)^\omega\,\mathsf{B}(\deg s_n))$ basic operations from K which simplifies to $O((n-2^j+1)^\omega\,\mathsf{B}(nd))$ operations from K. The shape of U, before Phase 3, is

$$U = \left[\begin{array}{c|c} * & \\ \hline * & I_{n-2^j+1} \end{array}\right].$$

Thus updating U and V is free and the cost of Phase 3 is just the cost of computing $U^{(j)}$ and $V^{(j)}$. Since $\deg U^{(j)}, \deg V^{(j)} < \deg s_{n_j}$, the cost of computing $U^{(j)}, V^{(j)}$ is bounded by $\mathsf{MM}(n,2^j,2^j)\mathsf{B}(\deg s_{n_j})$ and $\mathsf{MM}(2^j,2^j,n)\mathsf{B}(\deg s_{n_j})$ respectively, which simplifies to $O((n2^{j(\omega-1)}\mathsf{B}(\deg s_{n_j}))$ operations from K. Using Theorem 1.4 this cost bound simplifies to $O((n2^{j(\omega-1)}\mathsf{B}(nd/2^j))$ basic operations from K. This can be further simplified using the superlinearity of B (1.9) and the fast multiplication assumption (1.6) to $O(n^\omega \log n\mathsf{B}(d))$ operations from K. Therefore, the cost of the algorithm is dominated by the cost of Phase 1 and Phase 2. The cost of the Algorithm is bounded by $O(n^\omega\,\mathsf{B}(nd))$ basic operations from K. The cost of first two phases is higher than the target cost of $O\tilde{}(n^\omega d)$. The next idea deals with reducing the cost of Phase 1, i.e., the Smith decomposition phase.

Notice that the smallest invariant of $B^{(j)}$ is s_n/s_{n-2^j+1}. By Theorem 1.4, we know that s_n/s_{n-2^j+1} must divide the entries of $B^{(j)}$ and since $\deg B^{(j)} < s_n$, $\deg(s_{n-2^j+1}/s_nB^{(j)}) < \deg s_{n-2^j+1}$. We also know that $\deg T_l^{(j)}, \deg T_r^{(j)} < \deg s_{n-2^j+1}$, and $s_{n-2^j+1}/s_nS^{(j)} = \mathrm{diag}(s_{n-2^j+1}/s_{n-2^j+1}, s_{n-2^j+1}/s_{n-2^j},\ldots,s_{n-2^j+1}/s_{n-2^{j+1}})$. Therefore

$$\begin{aligned} & B^{(j)}_{[1,2,\ldots,2^j|1,2,\ldots,2^j]} && = T_l^{(j)}S^{(j)}T_r^{(j)} \in (\mathsf{K}[x]/\langle s_n\rangle)^{2^j\times 2^j} \\ \Leftrightarrow\quad & \tfrac{s_{n-2^j+1}}{s_n}B^{(j)}_{[1,2,\ldots,2^j|1,2,\ldots,2^j]} && = T_l^{(j)}\left(\frac{s_{n-2^j+1}}{s_n}S^{(j)}\right)T_r^{(j)} \in (\mathsf{K}[x]/\langle s_{n-2^j+1}\rangle)^{2^j\times 2^j}. \end{aligned}$$

The idea is to reduce the entries of $B^{(j)}$ by pulling out the greatest common factor (the smallest invariant factor) and find a Smith decomposition of $B^{(j)}$ working over the ring

$\mathsf{K}[x]/\langle s_{n-2^j+1}\rangle$. Now the cost of Phase 1 becomes $O(2^{j\omega}\,\mathsf{B}(\deg s_{n-2^j+1}))$. Using Theorem 1.4 the cost is bounded by $O(2^{j\omega}\,\mathsf{B}(nd/2^j))$. Using superlinearity of B (1.9), the bound on cost of B (1.8) and the fast multiplication assumption (1.6), the cost of Phase 1 is bounded by $O(n^\omega\,(\log n)\mathsf{B}(d))$ basic operations from K.

Thus the algorithm now becomes:

$B^{(0)}, U, V = \overline{s_n A^{-1}}, I_n, I_n \in \mathsf{R}^{n\times n}$
for j **from** 0 **to** t **do**

$\mathcal{J} = 1, 2, \ldots, 2^j$
$n_j = n - 2^j + 1$

[1. Find the Smith Decomposition]
$S^{(j)} = \mathrm{diag}(s_{n-2^j+1}/s_{n-2^j+1}, s_{n-2^j+1}/s_{n-2^j}, \ldots, s_{n-2^j+1}/s_{n-2^{j+1}})$
$T_l^{(j)}, W, T_r^{(j)} = \mathtt{SD}(B^{(j)}_{[\mathcal{J}|\mathcal{J}]}, S^{(j)}, 2^j, s_{n_j}) \in (\mathsf{K}[x]/\langle s_{n_j}\rangle)^{2^j\times 2^j}$;

[2. Eliminate rows and columns]
$C^{(j)} := \mathrm{diag}((T_l^{(j)})^{-1}, I_{n-2^{j+1}+1}) B^{(j)} \mathrm{diag}((T_r^{(j)})^{-1}, I_{n-2^{j+1}+1}) \in \mathsf{R}^{n_j\times n_j}$;
$\left[\begin{array}{c|c} S^{(j)} & \\ \hline & B^{(j+1)} \end{array}\right] := \left[\begin{array}{c|c} I_{2^j} & \\ \hline C^{(j)}_{[2^j+1,\ldots,n-1|\mathcal{J}]}(C^{(j)}_{[\mathcal{J}|\mathcal{J}]})^{-1} & I_{n-2^j+1} \end{array}\right] C^{(j)} \left[\begin{array}{c|c} I_{2^j} & (C^{(j)}_{[\mathcal{J}|\mathcal{J}]})^{-1} C^{(j)}_{[\mathcal{J}|2^j+1,\ldots,n-1]} \\ \hline & I_{n-2^j+1} \end{array}\right]$;
$B^{(j+1)} := \frac{s_{n-2^{j+1}+1}}{s_{n-2^j+1}} B^{(j+1)}$;

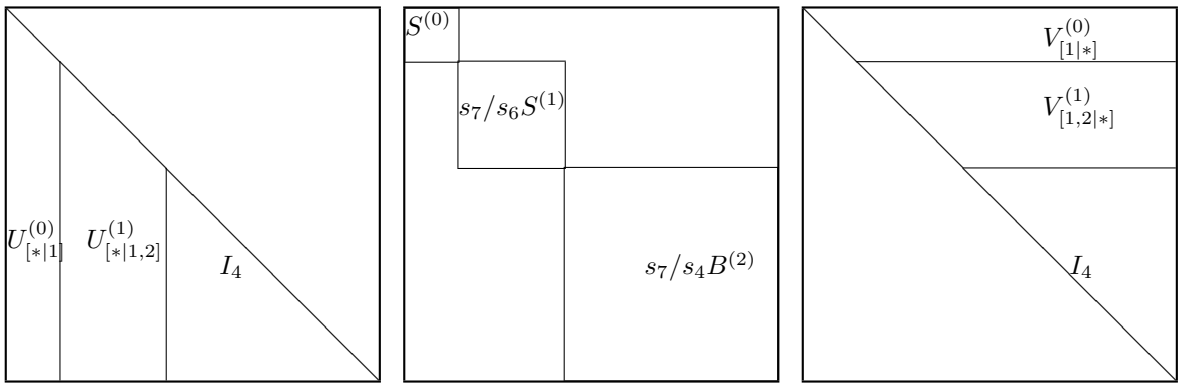
[3. Update]
$U^{(j)} := \mathrm{Rem}\left(\left[\begin{array}{c|c} T_l^{(j)} & \\ \hline -C^{(j)}_{[2^j+1,\ldots,n-1|\mathcal{J}]}(C^{(j)}_{[\mathcal{J}|\mathcal{J}]})^{-1} T_l^{(j)} & I_{n-2^{j+1}+1} \end{array}\right], s_{n_j}\right) \in \mathsf{R}^{n_j\times n_j}$;
$V^{(j)} := \mathrm{Rem}\left(\left[\begin{array}{c|c} T_r^{(j)} & -T_r^{(j)}(C^{(j)}_{[\mathcal{J}|\mathcal{J}]})^{-1} C^{(j)}_{[\mathcal{J}|2^j+1,\ldots,n-1]} \\ \hline & I_{n-2^{j+1}+1} \end{array}\right], s_{n_j}\right) \in \mathsf{R}^{n_j\times n_j}$;
$U := U \left[\begin{array}{c|c} I_{2^j-1} & \\ \hline & U^{(j)} \end{array}\right] \in \mathsf{R}^{n\times n}$;
$V := \left[\begin{array}{c|c} I_{2^j-1} & \\ \hline & V^{(j)} \end{array}\right] V \in \mathsf{R}^{n\times n}$;
Assert: $s_n A^{-1} = U\mathrm{diag}(s_n/s_n S^{(0)}, \ldots, s_n/s_{n-2^j+1} S^{(j)}, s_n/s_{n-2^{j+1}+1} B^{(j+1)}) V \mod s_n$
od

The cost Phase 2, i.e., the Eliminate rows and columns phase, still remains higher than the target cost of $O\tilde{}(n^\omega d)$ basic operations from K. The basic problem in reducing the cost of Phase 2 is that the size of $B^{(j)}$ can be potentially $O(n^3 d)$ in terms of number of terms from K. Subsection 3.2.2 develops new ideas to reduce this cost.

3.2.2 Delayed Updates: the Outer Product Formula

For nonsingular $A \in \mathsf{K}[x]^{n\times n}$ with $\deg A \le d$, we cannot work explicitly with s_nA^{-1} to compute a s_n-reduced Smith decomposition, $(U, \mathrm{snf}(s_nA^{-1}), V)$, of s_nA^{-1} in cost bounded by $O^{\sim}(n^{\omega}d)$, as the size of s_nA^{-1} can potentially be in the order of $O(n^3d)$ elements from K. This is the basic hurdle faced by algorithm in Subsection 3.2.1 to achieve the $O^{\sim}(n^{\omega}d)$ cost bound. This subsection discusses an idea which provides a way out and improves on the algorithm discussed in Subsection 3.2.1 to achieve the target cost. The special shape of U and V will be helpful in this cause.

Consider the algorithm developed in the end of Subsection 3.2.1. The following is the shape of U, $\mathrm{diag}(S^{(0)}, s_7/s_6S^{(1)}.s_7/s_4B^{(2)})$ and V after the 1-st iteration on a matrix $A \in \mathsf{R}^{7\times 7}$. Note that the iterations are numbered from 0.



Due to the structure of U, V and $\mathrm{diag}(S^{(0)}, \ldots, s_n/s_{n-2^j+1}S^{(j)}, s_n/s_{n-2^{j+1}+1}B^{(j+1)})$ we can modify our assertion. Let

$$\bar{U}^{(j)} = \left[\begin{array}{c} 0 \\ \hline U^{(j)}_{[*|1,2,\ldots,2^j]} \end{array}\right] \in \mathsf{R}^{n\times 2^j}, \bar{V}^{(j)} = \left[\begin{array}{c|c} 0 & V^{(j)}_{[1,2,\ldots,2^j|*]} \end{array}\right] \in \mathsf{R}^{2^j\times n}.$$

After j-th iteration, we can assert

$$\overline{s_nA^{-1}} = \overbrace{\left(\sum_{i=0}^{j} s_n/s_{n-2^i+1}\bar{U}^{(i)}S^{(i)}\bar{V}^{(i)}\right)}^{P^{(j)}} + s_n/s_{n-2^{j+1}+1}\left[\begin{array}{c|c} 0 & \\ \hline & B^{(j+1)} \end{array}\right] \in \mathsf{R}^{n\times n}. \qquad (3.11)$$

Let

$$P^{(j)} = \sum_{i=0}^{j} s_n/s_{n-2^i+1}\bar{U}^{(i)}S^{(i)}\bar{V}^{(i)} \mod s_n \in \mathsf{R}^{n\times n}. \qquad (3.12)$$

Rearranging the terms in (3.11) we get

$$\left[\begin{array}{c|c} 0 & \\ \hline & B^{(j+1)} \end{array}\right] = \frac{s_{n-2^{j+1}+1}}{s_n}\left(\overline{s_n A^{-1}} - P^{(j)}\right) \in \mathsf{R}^{n\times n}. \tag{3.13}$$

Further observe that to compute $\bar{U}^{(j+1)}$ and $\bar{V}^{(j+1)}$, the algorithm needs only $B_r^{(j+1)} = \mathrm{Rem}(B^{(j+1)}_{[1,2,\ldots,2^j|*]}, s_{n-2^j+1})$ and $B_c^{(j+1)} = \mathrm{Rem}(B^{(j+1)}_{[*|1,2,\ldots,2^j]}, s_{n-2^j+1})$. This can be done efficiently in cost bounded by $O(n^\omega \log n \mathsf{M}(d))$ operations from K. See Section 3.2.3 for details. Figure 3.2 describes the algorithm to get a reduced Smith transform of A or a reduced Smith Decomposition of $s_n A^{-1}$ (Lemma 3.3).

For the j-th iteration, the cost of Computing $B_c^{(j)}$ and $B_r^{(j)}$ phase is bounded by $O(n^\omega \log n\, \mathsf{M}(d))$, using Lemma 3.8. As proved in the end of the last section, the cost of Smith decomposition phase is bounded by $O(n^\omega(\log n)\mathsf{B}(d))$ basic operations from K. The cost of phase 3 and phase 4 is bounded by the cost of multiplications of matrices of dimensions $2^j \times 2^j$ and $2^j \times n$, where every term is reduced modulo s_{n-2^j+1}. Therefore the cost is bounded by $O(n/2^j\, 2^{j\omega}\mathsf{B}(\deg s_{n-2^j+1}))$. Using Theorem 1.4, fast multiplication assumption (1.6), and superlinearity of B (1.9), we can simplify this bound to $O(n^\omega(\log n)\,\mathsf{B}(d))$. We can now state the following result

Theorem 3.6. *Algorithm* `RST` *in Figure 3.2 is correct. Its cost is bounded by* $O(n^\omega(\log n)^2\,\mathsf{B}(d))$ *basic operations from* K. *We assume* $\mathsf{M}(t) \in O(t^{\omega-1})$ *for* $t \in \mathbb{Z}_{>0}$.

3.2.3 Computation of $B_c^{(j)}$ and $B_r^{(j)}$

In this subsection we show how to efficiently compute $B_r^{(j)} = \mathrm{Rem}(B^{(j)}_{[1,2,\ldots,2^j|*]}, s_{n-2^j+1})$ and $B_c^{(j)} = \mathrm{Rem}(B^{(j)}_{[*|1,2,\ldots,2^j]}, s_{n-2^j+1})$. We calculate these quantities over $\mathsf{K}[x]$ and then reduce them over R using ϕ_{s_n}. For this subsection, we avoid the use of ϕ to keep the presentation simple. Whenever for an object a over R, we state a over $\mathsf{K}[x]$ we imply $\phi_{s_n}^{-1}(a)$ over $\mathsf{K}[x]$. Using (3.13),

$$\left[\begin{array}{c|c} 0 & \\ \hline & B^{(j)} \end{array}\right] \equiv \frac{s_{n-2^j+1}}{s_n}\left(s_n A^{-1} - P^{(j-1)}\right) \mod s_n \in \mathsf{K}[x]^{n\times n}$$

$$\Rightarrow \quad \left[\,0\middle|B_r^{(j)}\,\right] \equiv \overbrace{\left[\,0|I_{2^j}|0\,\right]}^{I_r^{(j)}}\left(\frac{s_{n-2^j+1}}{s_n}\left(s_n A^{-1} - P^{(j-1)}\right) \mod s_n\right) \mod s_{n-2^j+1} \in \mathsf{K}[x]^{2^j\times n} \tag{3.14}$$

$\texttt{RST}(A, S)$
Input: Nonsingular $A \in \mathsf{K}[x]^{n\times n}$, Smith form $S \in \mathsf{K}[x]^{n\times n}$ of A, with $d = \deg A$.

$\mathsf{R} = \mathsf{K}[x]/\langle s_n\rangle$.
$\overline{s_nA^{-1}} = \phi_{s_n}(s_nA^{-1})$.
Condition: • $\deg s_nA^{-1} < \deg s_n$.
• $\operatorname{snf}(\overline{s_nA^{-1}}_{[1,\ldots,i|i,\ldots,i]}) = \operatorname{diag}\left(\frac{s_n}{s_n}, \frac{s_n}{s_{n-1}}, \ldots, \frac{s_n}{s_{n-i+1}}\right) \in \mathsf{R}^{i\times i}, 1 \le i \le n.$

Output: $(U, V) \in \mathsf{K}[x]^{n\times n}$, a reduced Smith transform of A or fail if the second condition is invalid.

[Working over R: Computing a reduced Smith Decomposition of $\overline{s_nA^{-1}}$]

$B^{(0)}, U, V = \overline{s_nA^{-1}}, I_n, I_n \in \mathsf{R}^{n\times n}$
for j **from** 0 **to** t **do**

[1. Compute B_r, B_c]
$\mathcal{J} = 1, 2, \ldots, 2^j$
$n_j = n - 2^j + 1$
$B_r^{(j)} := \operatorname{Rem}(\left[\begin{array}{c|c} I_{2^j} & 0 \end{array}\right] B^{(j)}, s_{n-2^j+1}) \in \mathsf{R}^{2^j\times n_j}$;
$B_c^{(j)} := \operatorname{Rem}(B^{(j)} \left[\begin{array}{c|c} I_{2^j} & 0 \end{array}\right]^T, s_{n-2^j+1}) \in \mathsf{R}^{n_j\times 2^j}$;

[2. Find the Smith Decomposition]
$S^{(j)} = \operatorname{diag}(s_{n-2^j+1}/s_{n-2^j+1}, s_{n-2^j+1}/s_{n-2^j}, \ldots, s_{n-2^j+1}/s_{n-2^{j+1}})$
$T_l^{(j)}, W, T_r^{(j)} = \texttt{SD}((B_r^{(j)})_{[\mathcal{J}|\mathcal{J}]}, S^{(j)}, 2^j, s_{n_j}) \in (\mathsf{K}[x]/\langle s_{n_j}\rangle)^{2^j\times 2^j}$;

[3. Eliminate rows and columns]
$C_r^{(j)} := \operatorname{Rem}((T_l^{(j)})^{-1}B_r^{(j)}, s_{n_j}) \in R^{2^j\times n}$;
$C_c^{(j)} := \operatorname{Rem}(B_c^{(j)}(T_r^{(j)})^{-1}, s_{n_j}) \in R^{n\times 2^j}$;

[4. Update]
$U^{(j)} := \operatorname{Rem}\left(\left[\begin{array}{c|c} T_l^{(j)} & \\ \hline -(C_c^{(j)})_{[2^j+1,\ldots,n-1|*]}((C_c^{(j)})_{[\mathcal{J}|*]})^{-1}T_l^{(j)} & I_{n-2^{j+1}+1} \end{array}\right], s_{n_j}\right) \in \mathsf{R}^{n_j\times n_j}$;
$V^{(j)} := \operatorname{Rem}\left(\left[\begin{array}{c|c} T_r^{(j)} & -T_r^{(j)}((C_r^{(j)})_{[*|\mathcal{J}]})^{-1}(C_r^{(j)})_{[*|2^j+1,\ldots,n-1]} \\ \hline & I_{n-2^{j+1}+1} \end{array}\right], s_{n_j}\right) \in \mathsf{R}^{n_j\times n_j}$;
$\bar{U}^{(j)} := \left[\begin{array}{c} O \\ \hline U^{(j)} \end{array}\right] \in R^{n\times 2^j}$;
$\bar{V}^{(j)} := \left[\begin{array}{c|c} 0 & V^{(j)} \end{array}\right] \in \mathsf{R}^{2^j\times n}$;
Assert: $\overline{s_nA^{-1}} = \left(\sum_{i=0}^{j} s_n/s_{n_i}\bar{U}^{(i)}S^{(i)}\bar{V}^{(i)}\right) + s_n/s_{n_{j+1}}\left[\begin{array}{c|c} 0 & \\ \hline & B^{(j+1)} \end{array}\right] \in R^{n\times n}$

od

$P \in \mathsf{K}[x]^{n\times n}$ be a Permutation matrix with 1 on the anti diagonal.
return $P\phi_{s_n}^{-1}\left(\left[\begin{array}{c|c|c|c} (\bar{V}^{(0)})^T & \cdots & (\bar{V}^{(t)})^T \end{array}\right]^T\right), \phi_{s_n}^{-1}\left(\left[\begin{array}{c|c|c} \bar{U}^{(0)} & \cdots & \bar{U}^{(t)} \end{array}\right]\right) P \in \mathsf{K}[x]^{n\times n}$;

Figure 3.2: `Reduced Smith Transform`

$$\left[\begin{array}{c} O \\ \hline B_c^{(j)} \end{array}\right] \equiv \left(\frac{s_{n-2^j+1}}{s_n} \left(s_n A^{-1} - P^{(j-1)} \right) \mod s_n \right) \overbrace{\left[\begin{array}{c} 0 \\ \hline I_{2^j} \\ \hline 0 \end{array}\right]}^{I_c^{(j)}} \mod s_{n-2^j+1} \in \mathsf{K}[x]^{n \times 2^j}. \tag{3.15}$$

The problem of computing $B_r^{(j)}$ and the problem of computing $B_c^{(j)}$ are similar. Thus let us consider computing $B_c^{(j)}$. Note that $I_c^{(j)}$ is a scalar matrix and hence

$$\left(\frac{s_{n-2^j+1}}{s_n} \left(s_n A^{-1} - P^{(j-1)} \right) \mod s_n \right) I_c^{(j)} \mod s_{n-2^j+1}$$

$$\equiv \left(\frac{s_{n-2^j+1}}{s_n} \left(s_n A^{-1} I_c^{(j)} - P^{(j-1)} I_c^{(j)} \right) \mod s_n \right) \mod s_{n-2^j+1}$$

Note that even though $s_{n-2^j+1}/s_n \left(s_n A^{-1} - P^{(j-1)} \right)$ is over $\mathsf{K}[x]$, $s_{n-2^j+1} A^{-1} I_c^j$ and $s_{n-2^j+1}/s_n P^{(j-1)} I_c^j$ may not be over $\mathsf{K}[x]$. Thus, to get $B_c^{(j)}$, we compute the series expansion of $\mathrm{Rem}(s_{n-2^j+1} A^{-1} I_c^j, s_n)$ and $\mathrm{Rem}(s_{n-2^j+1}/s_n P^{(j-1)} I_c^{(j)}, s_n)$ over $\mathsf{K}[[x]]$ up to a required precision.

To bound the precision, as in [30], we make use of the following fact.

Lemma 3.7. *Let $a, N \in \mathsf{K}[x]$ with N nonzero. Then $a = \mathrm{Rem}(a, N)$ if $\deg N \geq \deg a + 1$.*

We can bound the degree of $s_{n-2^j+1}/s_n \left(s_n A^{-1} I_c^{(j)} - P^{(j-1)} I_c^{(j)} \right)$, using (3.7), (3.12) and Theorem 1.4,:

$$\begin{aligned} \deg \frac{s_{n-2^j+1}}{s_n} \left(s_n A^{-1} I_c^{(j)} - P^{(j-1)} I_c^{(j)} \right) &\leq \deg s_n - \deg s_n + \deg s_{n-2^j+1} \\ &= \deg s_{n-2^j+1} \leq \tfrac{nd}{2^j}. \end{aligned}$$

Using Lemma 3.7, we get

$$\frac{s_{n-2^j+1}}{s_n} \left(s_n A^{-1} I_c^{(j)} - P^{(j-1)} I_c^{(j)} \right)$$

$$= \mathrm{Rem} \left(s_{n-2^j+1} \overbrace{\mathrm{Rem}(A^{-1} I_c^j, x^{nd/2^j})}^{W_1} - s_{n-2^j+1}/s_n \overbrace{\mathrm{Rem} \left(\mathrm{Rem}(P^{(j-1)} I_c^{(j)}, s_n), x^{nd/2^j} \right)}^{W_2}, x^{nd/2^j} \right).$$

and as $s_{n-2^j+1} \mid s_n$, we get that

$$\mathrm{Rem} \left(\mathrm{Rem} \left(\frac{s_{n-2^j+1}}{s_n} \left(s_n A^{-1} I_c^{(j)} - P^{(j-1)} I_c^{(j)} \right), s_n \right), s_{n-2^j+1} \right)$$

$$= \text{Rem}\left(\text{Rem}\left(s_{n-2^j+1}W_1 - s_{n-2^j+1}/s_n W_2, x^{nd/2^j}\right), s_{n-2^j+1}\right).$$

W_1 can be computed in cost $O(n^\omega \log n \mathsf{M}(d))$ using the high order-lifting techniques in [28, Section 9]. For the computation of W_2 we use the techniques developed in Section 2.3. Algorithm 3.3 describes how we compute W_2.

From (3.12), we know:

$$P^{(j-1)} \equiv \sum_{i=0}^{j-1} s_n/s_{n-2^i+1}\bar{U}^{(i)}S^{(i)}\bar{V}^{(i)} \mod s_n \in \mathsf{K}[x]^{n\times n}.$$

Therefore

$$\begin{aligned}\text{Rem}\left(P^{(j-1)}I_c^{(j)}, s_n\right) &= \textstyle\sum_{i=0}^{j-1} \text{Rem}\left(s_n/s_{n-2^i+1}\bar{U}^{(i)}S^{(i)}\bar{V}^{(i)}I_c^{(j)}, s_n\right)\\ &= \textstyle\sum_{i=0}^{j-1} s_n/s_{n-2^i+1}\text{Rem}\left(\overbrace{\bar{U}^{(i)}S^{(i)}}^{U_s}\overbrace{\bar{V}^{(i)}I_c^{(j)}}^{V_s}, s_{n-2^i+1}\right).\end{aligned}$$

This implies

$$\begin{aligned}W_2 &= \text{Rem}\left(\text{Rem}(P^{(j-1)}I_c^{(j)}, s_n), x^{nd/2^j}\right)\\ &= \textstyle\sum_{i=0}^{j-1} \text{Rem}\left(\text{Rem}\left(U_sV_s, s_{n-2^i+1}\right), x^{nd/2^j}\right)\end{aligned}$$

In iteration i of Algorithm 3.3, we compute $\text{Rem}\left(\text{Rem}\left(U_sV_s, s_{n-2^i+1}\right), x^{nd/2^j}\right)$ by invoking Algorithm 2.4. Recall $\bar{U}^{(i)} \in \mathsf{K}[x]^{n\times 2^i}, \bar{V}^{(i)} \in \mathsf{K}[x]^{2^i\times n}, S^{(i)} \in \mathsf{K}(x)^{2^i\times 2^i}$. Computing U_s is just multiplying every entry in the k-th column of $\bar{U}^{(i)}$ by $S^{(i)}_{i,i}$, for $0 < i \leq 2^i$. Since every entry in $S^{(i)}$ is a proper fraction or scalar, $\deg U_s < \deg \bar{U}^{(i)}$. The cost of computing U_s is bounded by $n2^i\mathsf{M}(\deg s_{n-2^i+1})$. Using Theorem 1.4, we can bound this cost by $O(n2^i\,\mathsf{M}(nd/2^i))$. This bound can be further simplified to $O(n\mathsf{M}(nd))$ by using (1.1) the superlinearity of M. The nonzero entries of V_s are read of from the corresponding entries in $\bar{V}^i$, thus it entails no cost. Lastly the cost of `PartialProduct` operation is bounded by $O(n\mathsf{M}(nd) + n^\omega\,\mathsf{M}(d))$, using Corollary 2.9. Thus the total cost of one iteration of Algorithm 3.3 is bounded by $O(n\mathsf{M}(nd) + n^\omega\,\mathsf{M}(d))$. Using the fast multiplication assumption (1.6), we can bound the cost of each iteration by $O(n^\omega\,\mathsf{M}(d))$. As j is bounded by $\log n$, the total cost of Algorithm 3.3 is bounded by $O(n^\omega(\log n)\,\mathsf{M}(d))$. We can state the following lemma:

Lemma 3.8. *$B_c^{(j)}$ and $B_r^{(j)}$ can be computed in $O(n^\omega(\log n)\,\mathsf{M}(d))$ basic field operations from K. We assume that $\mathsf{M}(t) \in O(t^{\omega-1})$ for $t \in \mathbb{Z}_{>0}$.*

Computing $\mathrm{Rem}(\mathrm{Rem}(P^{(j-1)}I_c^{(j)}, s_n), x^{nd/2^j})$
Input: $s_n, \left(s_{n-2^i+1}, \bar{U}^{(i)}, S^{(i)}, \bar{V}^{(i)}\right)_{0\le i\le j-1}, Y, N$
Output: $\mathrm{Rem}(\mathrm{Rem}(s_{n-2^j+1}/s_n P^{(j-1)}I_c^{(j)}, s_n), x^{nd/2^j}) \in \mathsf{K}[[x]]^{n\times 2^j}$

for $i = 0$ to $j-1$
  Comment *Calculating* $\mathrm{Rem}(\mathrm{Rem}(s_n/s_{n-2^i+1}\bar{U}^{(i)}S^{(i)}\bar{V}^{(i)}I_c^{(j)}, s_n), x^{nd/2^j})$
  $U_s := Rem(\bar{U}^{(i)}S^{(i)}, s_{n-2^i+1}) \in \mathsf{K}[x]^{n\times 2^i}$;
  $V_s := \bar{V}^{(i)}I_c^{(i)} \in \mathsf{K}[x]^{2^i\times n}$;
  $\tau_{k-j} := \texttt{PartialProduct}(s_n/s_{n-2^i+1}, U_s, V_s, s_n, \lceil nd/2^j\rceil) \in \mathsf{K}[[x]]^{n\times n}$
od
return $\sum_{j=0}^{i-1}\tau_{k-j}$;

Figure 3.3: Computing $\mathrm{Rem}(\mathrm{Rem}(P^{(j-1)}I_c^{(j)}, s_n), x^{nd/2^j})$

Chapter 4

Diagonal Entries of the Hermite Form

Throughout this chapter let $A \in \mathsf{K}[x]^{n \times n}$ be nonsingular with degree d.

In this chapter we show how to pass over the Smith form of A in order to recover the degrees of the diagonal entries of the Hermite form of A. The algorithm actually recovers the diagonal entries and not just the degrees, but it is the degrees that will be required by our Hermite form algorithm in the next chapter. Section 4.1 details a way of efficiently computing the Hermite form of a matrix where the largest invariant factor of the input matrix is small. Some mathematical background and previous results are developed and recalled in Section 4.2. Finally, the algorithm for computing the diagonal entries is given in Section 4.3.

4.1 Hermite form via Howell form

The classical method to compute the Hermite form is to work modulo the determinant [8, 7, 16, 17, 14]. Given $\operatorname{Rem}(A, \det A)$ and $\det A$, this leads to an $O(n^\omega \, \mathsf{B}(\deg \det A))$ algorithm for computing the Hermite form. For some input matrices, though, the largest Smith invariant factor may be much smaller in degree than the determinant of the matrix. The following example shows a matrix of the type we might expect to encounter in our algorithm to compute the diagonal entries of the Hermite form. Note that the largest invariant factor

of the matrix is x^d, while the the determinant of the matrix is x^{nd}.

$$\left[\begin{array}{cccc|cccc} x^d & & & & & & & \\ & x^d & & & & & & \\ & & \ddots & & & & & \\ & & & x^d & & & & \\ \hline * & * & \cdots & * & 1 & * & \cdots & * \\ * & * & \cdots & * & & 1 & & * \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & \vdots \\ * & * & \cdots & * & & & & 1 \end{array}\right] \in \mathsf{K}[x]^{2n\times 2n}$$

To make the cost of computing the Hermite form sensitive to the largest invariant factor, we work with precision just enough to contain the largest invariant factor. For nonsingular $A \in \mathsf{K}[x]^{n\times n}$, let s_n be the largest invariant factor. We will work over $\mathsf{R} = \mathsf{K}[x]/\langle x s_n\rangle$ (see Section 1.2 for details on the proscribed set of associates and residues in R, and the basic operations in R). The ring R is not a PID because of the presence of zero divisors and hence the Hermite form is not a canonical form over R. For example both of the following matrices are in Hermite form over $\mathbb{Z}_7[x]/\langle x^3+2x^2\rangle$, and they are left equivalent over R. The largest Smith invariant of the matrix on the left is x^2+5x+6 over $\mathsf{K}[x]$.

$$\begin{bmatrix} x+2 & 1 \\ & x+3 \end{bmatrix} \equiv_L \begin{bmatrix} x+2 & 1 \\ & x^2+5x+6 \end{bmatrix} \in (\mathbb{Z}_7[x]/\langle x^3+5x^2+6x\rangle)^{2\times 2}.$$

We need to introduce some notions over R, which will be helpful in the computation of the Hermite form of $A \in \mathsf{K}[x]^{n\times n}$.

Let R be any principal ideal ring. For a matrix $A \in \mathsf{R}^{n\times m}$ we write $S(A)$ to mean the set of all R-linear combinations of rows of A and $S_j(A)$ to mean the subset of $S(A)$ comprised of all rows which have first j entries zero. We can now define the Howell form over R.

Definition 4.1. *Corresponding to every $A \in \mathsf{R}^{*\times m}$ is an $H \in \mathsf{R}^{*\times m}$ that satisfies:*

1. *Let r be the number of nonzero rows of H. Then the first r rows of H are nonzero. For $0 \le i \le r$ let $H[i, j_i]$ be the first non zero entry in row i. Then $0 = j_0 < j_1 < j_2 < \cdots < j_r$.*
2. *$H[i, j_i] \in \mathcal{A}(\mathsf{R})$ and $H[k, j_i] \in \mathcal{R}(\mathsf{R}, H[i, j_i])$ for $1 \le k < i \le r$.*
3. *Rows $i+1, i+2, \ldots, r$ of H generate $S_{j_i}(A)$.*

H is the Howell canonical form of A [15].

For the rest of this section let $\mathsf{R} = \mathsf{K}[x]/\langle xs \rangle$. Recall from Section 1.2 that $\phi_{xs_n} : \mathsf{K}[x] \to \mathsf{R}$ is the homomorphism $\phi_{xs_n}(a) = \mathrm{Rem}(a, xs_n)$, for $a \in \mathsf{K}[x]$. For any $b \in \mathsf{R}$, we defined $\phi^{-1}_{xs_n}(b)$ as the unique element of $\mathcal{R}(\mathsf{K}[x], xs_n)$ that maps to b under ϕ. For example if $b = 3x + 1 \in \mathsf{K}[x]/\langle x^3 + x \rangle$, then $\phi^{-1}_{x^3+x}(3x + 1) = 3x + 1 \in \mathsf{K}[x]$. For a matrix $A \in \mathsf{K}[x]^{n\times m}$, we write $\mathcal{L}(A)$ to mean the set of all $\mathsf{K}[x]$-linear combination of rows of A. The next two lemmas show how we can compute the Hermite form of a nonsingular $A \in \mathsf{K}[x]^{n\times n}$, by passing over the Howell form of $\phi_{xs_n}(A)$ over $\mathsf{R} = \mathsf{K}[x]/\langle xs_n \rangle$.

Lemma 4.2. *Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular with Hermite form*

$$H = \begin{bmatrix} h_1 & \bar{h}_{12} & \cdots & \bar{h}_{1n} \\ & h_2 & \cdots & \bar{h}_{2n} \\ & & \ddots & \vdots \\ & & & h_n \end{bmatrix}$$

and Smith form

$$S = \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_n \end{bmatrix}.$$

Then $h_i \mid s_n$ for $1 \le i \le n$. Furthermore, for any $s \in \mathsf{K}[x]$ we have $s_n \mid s$ if and only if $sI_n \in \mathcal{L}(A)$.

Proof. Let s_i^* be the gcd of all $i \times i$ minors of A, where s_0^* is defined to be 1. Let h_i^* be the gcd of all $i \times i$ minors of $A_{[*|1,2,\ldots,i]}$, where h_0^* is defined to be 1. We know from Theorem 1.4 and Theorem 1.2 that $s_i = s_i^*/s_{i-1}^*$ and $h_i = h_i^*/h_{i-1}^*$. Note $h_n^* = s_n^*$, therefore $h_n \mid s_n$. Hence $s_n I[n, *] \in \mathcal{L}(A)$. We can swap columns i and n in A and repeat the whole argument and swap the columns i and n back to get that $h_i \mid s_n$ and $s_n I[i, *] \in \mathcal{L}(A)$. Therefore $s_n I \in \mathcal{L}(A)$ and hence $sI \in \mathcal{L}(A)$ if $s_n \mid s$.

To complete the proof we prove that if $sI_n \in \mathcal{L}(A)$, then $s_n \mid s$. We know that $sI_n \in \mathcal{L}(A)$, which implies that $sI_n = U'A$, for a nonsingular $U' \in \mathsf{K}[x]^{n\times n}$. Since $\mathrm{snf}(A) = S$, $A = USV$, for some unimodular $U, V \in \mathsf{K}[x]^{n\times n}$. Therefore

$$\begin{aligned} & sI_n && = U'A \\ \Leftrightarrow \quad & sI_n && = U'USV \\ \Leftrightarrow \quad & sV^{-1} && = U'US \\ \Leftrightarrow \quad & sI_n && = VU'US, \end{aligned}$$

where $VU'U$ is nonsingular over $\mathsf{K}[x]$, hence $sI_n \in \mathcal{L}(S)$, thus $s_n | s$. □

Lemma 4.3. *For nonsingular $A \in \mathsf{K}[x]^{n\times n}$ and nonzero $s \in \mathsf{K}[x]$ such that $sI_n \in \mathcal{L}(A)$, we have*

$$\mathit{Hermite}(A) = \phi_{xs}^{-1}(\mathit{Howell}(\phi_{xs}(A))). \tag{4.1}$$

Proof. Let H be the Hermite form of A over $\mathsf{K}[x]$ and let R be the residue class ring $\mathsf{R} = \mathsf{K}[x]/\langle xs\rangle$. To prove the result it will suffice to show that $\bar{H} = \phi_{xs}(H) \in \mathsf{R}^{n\times n}$ is in Howell form over R.

The definition of the Howell form is given in Definition 4.1. Note that the first condition of the Howell form is satisfied by $\bar{H}$ by definition of Hermite form given in Definition 1.1 as $\bar{H}$ is upper triangular. For $\bar{H}$ to satisfy the second condition of Definition 4.1 it is enough to show that the diagonal entries of H are divisors of xs. Since $xs \in \mathcal{L}(A)$, by Lemma 4.2 the diagonal entries of H divide xs.

We shall now show that $\bar{H}$ obeys the third property as well, i.e., rows $i+1, i+2, \ldots, n$ of $\bar{H}$ generate $S_i(A)$ over R. We now prove the third property for $i = 1$. Let $\bar{r} \in S_1(A)$:

$$\begin{aligned} \bar{r} &= \bar{a}_1\bar{H}[1,*] + \bar{a}_2\bar{H}[2,*] + \cdots + \bar{a}_n\bar{H}[n,*] \in S_1(A) \text{ over } \mathsf{R}, \\ r &= a_1H[1,*] + a_2H[2,*] + \cdots + a_nH[n,*] \in \mathsf{K}[x], \end{aligned} \tag{4.2}$$

where $a_i = \phi_{xs}^{-1}(\bar{a}_i)$, $1 \le i \le n$. Therefore

$$\bar{r} = \phi_{xs}(r)$$

Since $\bar{r}[1] = 0$, by definition of $S_1(A)$, $\bar{a}_1$ is an annihilator of $\bar{H}[1,1]$, i.e., $\bar{a}_1\bar{H}[1,1] = 0 \in \mathsf{R}$. This implies that for some $c \in \mathsf{K}[x]$,

$$a_1H[1,1] = c(xs) \in \mathsf{K}[x].$$

We know that $sI[1,*] \in \mathcal{L}(A)$ over $\mathsf{K}[x]$. Thus

$$cxsI[1,*] = \begin{bmatrix} a_1H[1,1] & 0 & \cdots & 0 \end{bmatrix} \in \mathcal{L}(A). \tag{4.3}$$

Hence subtracting 4.3 from 4.2, we get $r' \in \mathcal{L}(A)$ with $r'[1] = 0$:

$$r' = r - csxI[1,*] \in S_1(A) \in \mathsf{K}[x]^{1\times n}.$$

Note that

$$\bar{r} = \phi_{xs}(r'). \tag{4.4}$$

Since over $\mathsf{K}[x]$ the Hermite form is also in a Howell form, rows $2, 3, \ldots, n$ of H can generate r', i.e.,

$$r' = b_2H[2,*] + b_3H[3,*] \cdots + b_nH[n,*] \in \mathsf{K}[x]$$

for some $b_i \in \mathsf{K}[x]$.

By (4.4), $\bar{r} = \phi_{xs}(r')$. Thus $\bar{r} \in S_1(A)$ over R can be generated by rows $2, 3, \ldots, n$ of $\bar{H}$. By induction on i, $1 \leq i \leq n$, we can argue that $S_i(A)$ is generated by the rows $i+1, i+2, \ldots, n$ of $\bar{H}$ over R for all $1 \leq i \leq n$. □

Using (4.1) and the result from [32, Section 4], [26, Proposition 4.8] on computation of Howell form we have the following result.

Theorem 4.4. *Given a nonsingular $A \in \mathsf{K}[x]^{n \times n}$ together with a nonzero $s \in \mathsf{K}[x]$ such that $sI \in \mathcal{L}(A)$, the Hermite form of A over $\mathsf{K}[x]$ can be computed in $O(n^\omega \, \mathsf{B}(\deg s))$ basic operations from K.*

We like to comment that for $s = xs_n$ all the conditions in Theorem 4.4 are satisfied, where s_n is the largest Smith invariant factor of A. The conditions in Theorem 4.4 are also satisfied for $s = x \det(A)$. Thus we can compute the Hermite form of a matrix via the Howell form in cost bounded by $O(n^\omega \, \mathsf{B}(nd))$. If we know s_n, then we can compute it in cost $O(n^\omega \, \mathsf{M}(\deg s_n) + n^2 \, \mathsf{B}(\deg s_n))$ basic operations from K. The $O(n^2 \, \mathsf{B}(\deg s_n))$ part of the cost is just to reduce all the entries of the input matrix modulo xs_n.

4.2 Hermite form via kernel basis

The Hermite form is a canonical form for left equivalence over $\mathsf{K}[x]$. Recall from Section 1.2 that a Hermite form H is *the Hermite form of A* if H is left equivalent to A: $H = UA$ for a unimodular transformation U. Solving for U gives $U = HA^{-1}$. The following lemma gives an alternative, equivalent criteria for a Hermite form H to be left equivalent to A that does not explicitly involve U.

Lemma 4.5. *A Hermite form H is the Hermite form of A if* $\deg \det H \leq \deg \det A$ *and HA^{-1} is over $\mathsf{K}[x]$.*

To obtain a more compact representation of the matrix A^{-1} in Lemma 4.5 we will pass over the Smith form. Recall from definition 1.3: corresponding to A are unimodular matrices $\bar{U}, \bar{V} \in \mathsf{K}[x]^{n \times n}$ such that $S = \bar{U}A\bar{V} = \mathrm{Diag}(s_1, \ldots, s_n)$ is the Smith canonical form of A, that is, each s_i is monic and $s_i \mid s_{i+1}$ for $1 \leq i \leq n-1$. Solving for A^{-1} gives $A^{-1} = \bar{V}S^{-1}\bar{U}$. Considering Lemma 4.5, and noting that $\bar{U}$ is unimodular, we may conclude that, for any matrix $H \in \mathsf{K}[x]^{n \times n}$, HA^{-1} is over $\mathsf{K}[x]$ if and only if $H\bar{V}S^{-1}$ is over $\mathsf{K}[x]$. Multiplying S^{-1} by s_n, the largest invariant factor, gives $s_nS^{-1} = \mathrm{Diag}(s_n/s_1, \ldots, s_n/s_n) \in \mathsf{K}[x]^{n \times n}$. We conclude that $H\bar{V}S^{-1}$ is over $\mathsf{K}[x]$ if and only if $H\bar{V}(s_nS^{-1}) \equiv 0 \bmod s_n$. We obtain the following result.

Lemma 4.6. *Suppose $S = \bar{U}A\bar{V} = \mathrm{Diag}(s_1, \ldots, s_n)$ is the Smith form of A, where $\bar{U}$ and $\bar{V}$ are unimodular, and let $V \in \mathsf{K}[x]^{n\times n}$ be the matrix obtained from $\bar{V}$ by reducing column j of $\bar{V}$ modulo s_j, $1 \le j \le n$. Then a Hermite form H is the Hermite form of A if* $\deg\det H \le \deg\det A$ *and* $HV(s_nS^{-1}) \equiv 0 \bmod s_n$.

The following corollary of Lemma 4.6 is the basis for our approach to compute the diagonal entries of the Hermite form of A.

Corollary 4.7. *Let V and S be as in Lemma 4.6. The Hermite form of*

$$\left[\begin{array}{c|c} S & \\ \hline V & I_n \end{array}\right] \in \mathsf{K}[x]^{2n\times 2n} \tag{4.5}$$

has the shape

$$\left[\begin{array}{c|c} I_n & * \\ \hline & H \end{array}\right] \in \mathsf{K}[x]^{2n\times 2n}, \tag{4.6}$$

where H is the Hermite form of A.

Proof. First note that

$$\left[\frac{S}{V}\right] \equiv_L \left[\frac{S}{\bar{V}}\right],$$

where V is a unimodular matrix. It follows that the principal $n \times n$ submatrix of the Hermite form of the matrix in (4.5) must be I_n. It remains to prove that H is the Hermite form of A. The unimodular transformation that transforms the matrix in (4.5) to its Hermite form in (4.6) must have the following shape:

$$\left[\begin{array}{c|c} * & * \\ \hline -HVS^{-1} & H \end{array}\right] \left[\begin{array}{c|c} S & \\ \hline V & I_n \end{array}\right] = \left[\begin{array}{c|c} I_n & * \\ \hline & H \end{array}\right].$$

The result follows as the last n rows $\left[\begin{array}{c|c} -HVS^{-1} & H \end{array}\right]$ of the transformation matrix are a left kernel basis for

$$\left[\frac{S}{V}\right].$$

□

Theorem 4.8. *Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular of degree d. If $\#\mathsf{K} \ge 8n^3d$, matrices S and V as in Lemma 4.6 can be computed in a Las Vegas fashion with an expected number of $O(n^\omega(\log n)^2\,\mathsf{B}(d))$ operations from K.*

Proof. First compute a row reduced form R of A using the algorithm of [11], or the deterministic variant in [12]. Then transform R into $R' = LRU$, where $L \in \mathsf{K}^{n\times n}$ is a unit lower triangular matrix and $U \in \mathsf{K}^{n\times n}$ is a unit upper triangular matrix with the non diagonal, non zero entries filled with elements from K, chosen uniformly at random. Note that R' is still row reduced. This ensures that with probability at least $1/2$ the preconditions of Algorithm `RST` in Figure 3.2 are satisfied. We can now use the `RST` algorithm to get (U', V'), a reduced Smith transform of R' in the allotted cost (Theorem 3.6). Finally we reverse the preconditioning to get $V = UV'$. □

4.3 The algorithm for diagonal entries

Corresponding to a nonsingular input matrix $A \in \mathsf{K}[x]^{n\times n}$ of degree d, let S and V be as in Lemma 4.6. Instead of working with the matrix in (4.5) it will be useful to reverse the columns of V. To this end, let P be the $n \times n$ permutation matrix with ones on the anti diagonal. Note that post multiplying a matrix by P reverses the order of the columns. Our input matrix has the shape

$$
\begin{aligned}
G &= \left[\begin{array}{c|c} P & \\ \hline & I \end{array}\right]\left[\begin{array}{c|c} S & \\ \hline V & I \end{array}\right]\left[\begin{array}{c|c} P & \\ \hline & I \end{array}\right] \\
&= \left[\begin{array}{cccc|cccc} s_n & & & & & & & \\ & s_{n-1} & & & & & & \\ & & \ddots & & & & & \\ & & & s_1 & & & & \\ \hline * & * & \cdots & * & 1 & & & \\ * & * & \cdots & * & & 1 & & \\ \vdots & \vdots & \cdots & \vdots & & & \ddots & \\ * & * & \cdots & * & & & & 1 \end{array}\right] \in \mathsf{K}[x]^{2n\times 2n},
\end{aligned} \tag{4.7}
$$

and satisfies the following properties:

1. $\mathrm{Diag}(s_1, \ldots, s_n)$ is the Smith form of A and hence satisfies $\sum_{j=1}^{n} \deg s_j = \deg \det A \leq nd$, where $d = \deg A$.

2. Off diagonal entries in column j of G have degree less than the diagonal entry in the same column, $1 \leq j \leq n$.

Our goal is to recover the last n diagonal entries of the Hermite form of G. The standard approach to triangularize G, without any regard to cost or concern for growth of degrees, is

to use extended gcd computations and unimodular row operations to zero out entries below the pivot entry in each column.

for j **from** 1 **to** $2n-1$ **do**
  for i **from** $j+1$ **to** $2n$ **do**
    $(g, s, t, u, v) := \mathrm{Gcdex}(G[j,j], G[i,j])$;
    $\begin{bmatrix} G[j,*] \\ G[i,*] \end{bmatrix} := \begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} G[j,*] \\ G[i,*] \end{bmatrix}$
  od
od

Note that, for $j = 1, 2 \ldots, n-1$, the first $n-j$ iterations of the inner loop do nothing since the principal $n \times n$ block of G remains upper triangular during the elimination; omitting these vacuous iterations, the following example shows how the shape of the work matrix changes as in the case $n = 3$:

$$\left[\begin{array}{ccc|ccc} s_3 & & & & & \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline * & * & * & 1 & & \\ * & * & * & & 1 & \\ * & * & * & & & 1 \end{array}\right] \to \left[\begin{array}{ccc|ccc} * & * & * & * & & \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline & * & * & * & & \\ * & * & * & & 1 & \\ * & * & * & & & 1 \end{array}\right] \to \left[\begin{array}{ccc|ccc} * & * & * & * & * & \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline & * & * & * & & \\ & * & * & * & * & \\ * & * & * & & & 1 \end{array}\right]$$

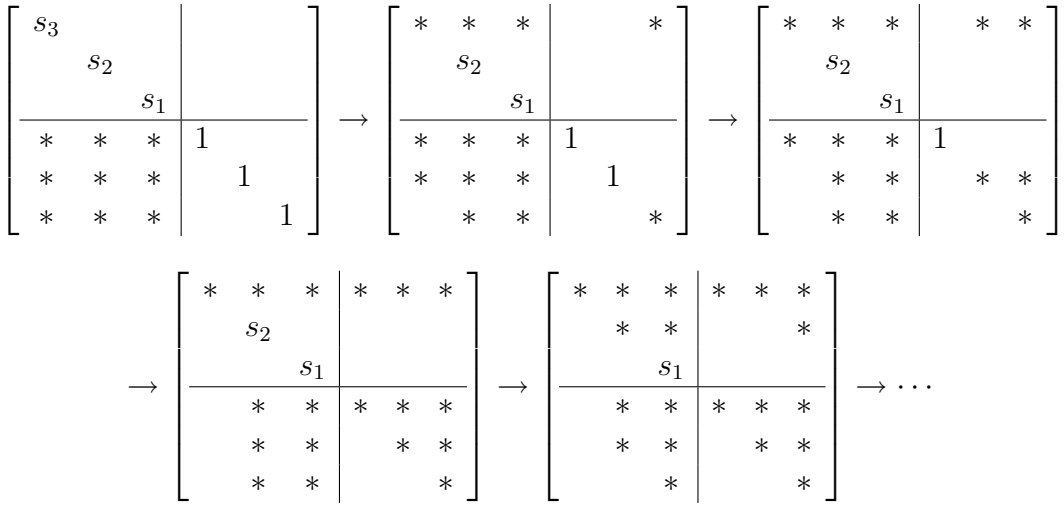
$$\to \left[\begin{array}{ccc|ccc} * & * & * & * & * & * \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline & * & * & * & & \\ & * & * & * & * & \\ & * & * & * & * & * \end{array}\right] \to \left[\begin{array}{ccc|ccc} * & * & * & * & * & * \\ & * & * & * & & \\ & & s_1 & & & \\ \hline & & * & * & & \\ & * & * & * & * & \\ & * & * & * & * & * \end{array}\right] \to \cdots$$

Note that even after only the first column has been eliminated, the upper triangular structure of the trailing $n \times n$ block has been lost, thus necessitating that j range up to $2n$. Our first refinement of the algorithm is to reverse the order of elimination of entries in the southwest block of G, thus preserving the upper triangularity of the southeast block.

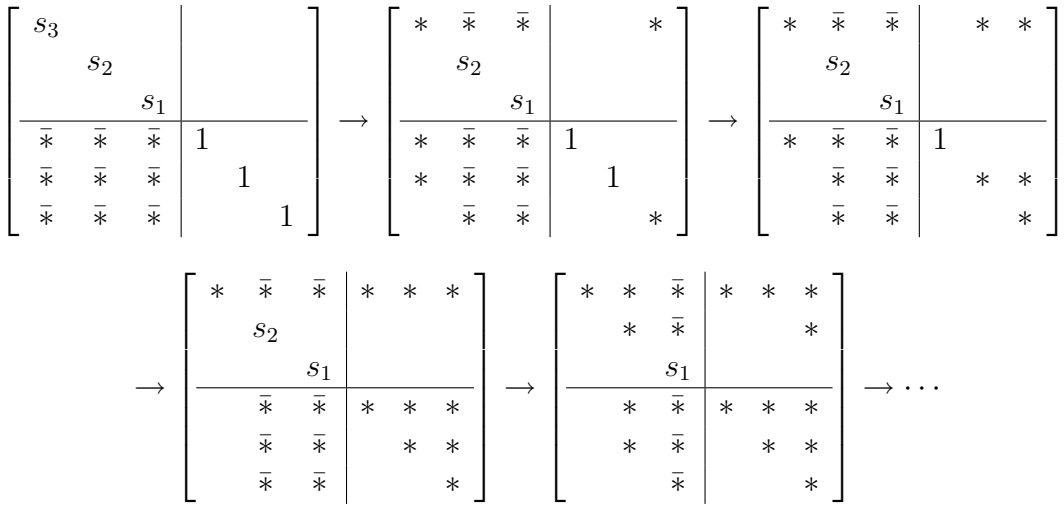
for j **from** 1 **to** n **do**
  for i **from** $2n$ **by** -1 **to** $n+1$ **do**
    $(g, s, t, u, v) := \mathrm{Gcdex}(G[j,j], G[i,j])$;
    $\begin{bmatrix} G[j,*] \\ G[i,*] \end{bmatrix} := \begin{bmatrix} s & t \\ u & v \end{bmatrix} \begin{bmatrix} G[j,*] \\ G[i,*] \end{bmatrix}$

od
od

The following example for $n = 3$ shows how the shape of the shape of the work matrix changes during the first few iterations:



Initially, we assume that the offdiagonal entries in G are reduced modulo the diagonal entry in the same column. As the algorithm eliminates entries in column j, we can implicitly perform unimodular row operations to reduce entries in column $j+1, \ldots, n$ modulo the diagonal entry in the same column. In the following example, entries that are kept reduced modulo the diagonal entry in the same column are represented by $\bar{*}$.



(4.8)

The second refinement of the algorithm is to keep the $\bar{*}$ entries reduced modulo the diagonal entry in the same column during the elimination.

for j **from** 1 **to** n **do**
  for i **from** $2n$ **by** -1 **to** $n+1$ **do**
    $(g, s, t, u, v) := \text{Gcdex}(G[j,j], G[i,j])$;
    $G[j,j], G[i,j] := g, 0$;
    $U := \begin{bmatrix} s & t \\ u & v \end{bmatrix}$;
    for k **from** $j+1$ **to** n **do**
      $\begin{bmatrix} G[j,k] \\ G[i,k] \end{bmatrix} := \text{Rem}\left(U \begin{bmatrix} G[j,k] \\ G[i,k] \end{bmatrix}, s_{n-k+1}\right)$
    od;
    for k **from** $n+1$ **to** $2n$ **do**
      $\begin{bmatrix} G[j,k] \\ G[i,k] \end{bmatrix} := U \begin{bmatrix} G[j,k] \\ G[i,k] \end{bmatrix}$
    od
  od
od

Notice in (4.8) that entries in the last n columns of the work matrix G are not kept reduced and can suffer from expression swell. However, our goal is to recover only the trailing n diagonal entries of the last n columns of the triangularization of G. To avoid the cost associated with performing the unimodular row operations on the last n columns of the work matrix, we can exploit the special structure of the work matrix and modify the elimination procedure to only keep track of the the last n diagonals. The following illustrates our point with an example for $n = 3$. Let

$$G = \left[\begin{array}{ccc|ccc} s_3 & & & & & \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline * & * & * & 1 & & \\ * & * & * & & 1 & \\ a_1 & * & * & & & 1 \end{array}\right].$$

The first elimination step computes the extended gcd of s_3 and a_1, $(g, s, t_1, u, v_1) = \mathrm{Gcdex}(s_3, a_1)$, and updates the work matrix to have the following shape:

$$\left[\begin{array}{ccc|ccc} s & & & & & t_1 \\ & 1 & & & & \\ & & 1 & & & \\ \hline & & & 1 & & \\ & & & & 1 & \\ u & & & & & v_1 \end{array}\right] \left[\begin{array}{ccc|ccc} s_3 & & & & & 0 \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline * & * & * & 1 & & \\ * & * & * & & 1 & \\ a_1 & * & * & & & 1 \end{array}\right] = \left[\begin{array}{ccc|ccc} g & * & * & & & t_1 \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline * & * & * & 1 & & \\ a_2 & * & * & & 1 & \\ & * & * & & & v_1 \end{array}\right].$$

Continuing the elimination gives

$$\left[\begin{array}{ccc|ccc} * & * & * & & & t_1 \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline * & * & * & 1 & & \\ a_2 & * & * & & 1 & \\ & * & * & & & v_1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} * & * & * & & t_2 & * \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline a_3 & * & * & 1 & & \\ & * & * & & v_2 & * \\ & * & * & & & v_1 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|ccc} * & * & * & t_3 & * & * \\ & s_2 & & & & \\ & & s_1 & & & \\ \hline & * & * & v_3 & * & * \\ & * & * & & v_2 & * \\ & a_4 & * & & & v_1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} * & * & * & * & * & * \\ & * & * & & & t_4 \\ & & s_1 & & & \\ \hline & * & * & v_3 & * & * \\ & * & * & & v_2 & * \\ & & * & & & v_1 v_4 \end{array}\right] \cdots$$

The key observation is that, although the offdiagonal entries in the last n columns are modified during the elimination, they never affect the last n diagonal entries entries which will depend only on the v_i computed by the calls to Gcdex. Our third refinement of the algorithm is to avoid storing and updating any of the offdiagonal entries in the last n columns of the matrix. Instead, we can keep track of the last n diagonal entries using a vector $D \in \mathsf{K}[x]^{1\times n}$. Thus, now we work on

$$G = \left[\begin{array}{c|c} P & \\ \hline & I \end{array}\right] \left[\begin{array}{c} S \\ \hline V \end{array}\right] P \in \mathsf{K}[x]^{2n\times n}, \tag{4.9}$$

instead of G in (4.7).

For $n = 3$, the following shows the state of G and D during the execution of Algorithm `DiagonalHermite` in Figure 4.1. Here v_i is the value of v on the i'th call to Gcdex in the

`DiagonalHermite`(S, V)

Input: • $S \in \mathsf{K}[x]^{n\times n}$, the Smith form of a nonsingular $A \in \mathsf{K}[x]^{n\times n}$ of degree d.
• $V \in \mathsf{K}[x]^{n\times n}$, $\deg \mathrm{Col}(V, j) < \deg s_j$, $1 \le j \le n$.

Output: $D \in \mathbb{Z}^{1\times n}$, the degrees of the last n diagonal entries in the Hermite form of $\left[\begin{array}{c|c} S & \\ \hline V & I \end{array}\right]$.

Let P be equal to I_n with columns reversed.
Initialize $G = \left[\begin{array}{c|c} P & \\ \hline & I \end{array}\right] \left[\begin{array}{c} S \\ \hline V \end{array}\right] P$.
Initialize $D = [1, \ldots, 1]$.
for j **from** 1 **to** n **do**
  for i **from** $2n$ **by** -1 **to** $n+1$ **do**
    $(g, s, t, u, v) := \mathrm{Gcdex}(G[j, j], G[i, j])$;
    $G[j, j], G[i, j] := g, 0$;
    $U := \begin{bmatrix} s & t \\ u & v \end{bmatrix}$;
    for k **from** $j+1$ **to** n **do**
      $U := \mathrm{Rem}(U, s_{n-k+1})$;
      $\begin{bmatrix} G[j, k] \\ G[i, k] \end{bmatrix} := \mathrm{Rem}\left(U \begin{bmatrix} G[j, k] \\ G[i, k] \end{bmatrix}, s_{n-k+1}\right)$
    od;
    $D[i-n] := D[i-n] \times v$
  od
od;
return $[\deg D[1], \ldots, \deg D[n]]$

Figure 4.1: Algorithm `DiagonalHermite`

above algorithm.

$$G = \left[\begin{array}{ccc} s_3 & & \\ & s_2 & \\ & & s_1 \\ \hline * & * & * \\ * & * & * \\ * & * & * \end{array}\right] \rightarrow \left[\begin{array}{ccc} * & * & * \\ & s_2 & \\ & & s_1 \\ \hline * & * & * \\ * & * & * \\ & * & * \end{array}\right] \cdots \rightarrow \left[\begin{array}{ccc} * & * & * \\ & s_2 & \\ & & s_1 \\ \hline & * & * \\ & * & * \\ & * & * \end{array}\right] \rightarrow \left[\begin{array}{ccc} * & * & * \\ & * & * \\ & & s_1 \\ \hline & * & * \\ & * & * \\ & & * \end{array}\right] \cdots$$

$$D = \begin{bmatrix} 1, & 1, & 1 \end{bmatrix} \qquad \begin{bmatrix} 1, & 1, & v_1 \end{bmatrix} \qquad \begin{bmatrix} v_3, & v_2, & v_1 \end{bmatrix} \qquad \begin{bmatrix} v_3, & v_2, & v_1 v_4 \end{bmatrix}$$

We now bound the running time of Algorithm `DiagonalHermite` in Figure 4.1. During the elimination of column j, entries in column j remain bounded in degree by the diagonal entry, a divisor of s_{n-j+1}. Thus, each call to Gcdex is bounded by $\mathsf{B}(\deg s_{n-j+1})$ operations from K. The cost of all n^2 calls to Gcdex is thus bounded by $n \sum_{j=1}^{n} \mathsf{B}(\deg s_j) \leq n\mathsf{B}(nd)$, using $\sum_{j=1}^{n} \deg s_j \leq nd$.

The cost of applying the transformation U, in each iteration of i, is bounded by $c_1 \sum_{k=1}^{n-j+1} \mathsf{M}(\deg s_k)$ for some constant $c_1 > 0$. For every column j, the total cost of applying transformations is bounded by $nc_1 \sum_{k=1}^{n-j+1} \mathsf{M}(\deg s_k)$. Thus the cost of applying the transformation U, in all iterations, is bounded by

$$nc_1 \sum_{j=1}^{n} \sum_{k=1}^{n-j+1} \mathsf{M}(\deg s_k) \leq c_1 n^2\, \mathsf{M}(nd),$$

using the superlinearity of M and that fact that $\sum_{j=1}^{n} \deg s_j \leq nd$.

Each entry in D is updated n times and also at any time during the execution of the algorithm $\sum_{i=1}^{n} \deg D[i] \leq nd$. This provides a bound for the cost of all updates to D as $O(n\mathsf{M}(nd))$.

We obtain the following result.

Theorem 4.9. *Algorithm* `DiagonalHermite` *in Figure 4.1 is correct. The cost of the algorithm is* $O(n^2\, \mathsf{M}(nd) + n\, \mathsf{B}(nd))$ *operations from* K.

4.3.1 Via matrix multiplication

In this section we adapt Algorithm `DiagonalHermite` in Figure 4.1 to a faster version using matrix multiplication. For the sake of simplicity assume that

$$n = \sum_{i=0}^{t} 2^i = 2^{t+1} - 1$$

This assumption can be made without loss of generality as we can always embed S and V into matrices $\mathrm{diag}(I, S)$ and $\mathrm{diag}(0, V)$ to get the required dimension and the new dimension shall at be at most twice of the original dimension.

We now introduce blocking. Figure 4.2 shows a block decomposition for a 14×7 matrix

$$G \;=\; \left[\begin{array}{c|c} P & \\ \hline & I \end{array}\right] \left[\begin{array}{c} S \\ \hline V \end{array}\right] P,$$

from (4.9), where $P \in \mathsf{K}[x]^{7\times 7}$ is a permutation matrix with 1 on the anti diagonal. The lower part is 7×7 matrix, we add a additional 0 row at the bottom for convenience of blocking.

Thus, for $G \in \mathsf{K}[x]^{2n\times n}$, instead of working on each entry one by one as in Algorithm `DiagonalHermite` in Figure 4.1, we shall split V, the lower part of G, into square blocks of sizes 2^i, for $i = 0, \ldots, t$. For keeping things simple, we pad V with an extra zero row to keep the dimension of V as $2^{t+1} \times (2^{t+1} - 1)$. We iterate on each block in the same order as we did in `DiagonalHermite` algorithm. For $n = 7$, Figure 4.2 shows the partition of the lower part of G. The numbers on the blocks are the order in which the algorithm iterates on the blocks.

Working over blocks instead of individual entries of the matrix, requires appropriate changes in the Eliminate, Apply Transformation and Update Diagonal steps of the algorithm. Figure 4.3 gives the basic scheme of `BlockDiagonalHermite` Algorithm. Figure 4.4 shows the shape of the matrix G before and after an iteration of `BlockDiagonalHermite` algorithm. The matrix on the left in Figure 4.4 is the shape of the matrix before Block Eliminate Phase and the matrix on the right is the shape of the matrix after Apply Block Transformation Phase. Non shaded area represents zero entries. The area boxed in the rectangle containing rows indexed by elements from S is the area affected by Apply Block Transformation Phase.

In Phase A, we compute the Hermite form H of W in cost $O(2^{j\omega}\,\mathsf{B}(\deg s_{n-2^j+1}))$ using the result in Theorem 4.4. To see that $s_{n-2^j+1} I \in \mathcal{L}(W)$, note $\mathrm{snf}(W) = \mathrm{diag}(I, \mathrm{snf}(G_{\mathcal{J},\mathcal{J}}))$. If we can show that s_{n-2^j+1} is a multiple of the largest invariant factor of $G_{\mathcal{J},\mathcal{J}}$, then by Lemma 4.2 we are done.

Initially $G^{(0)}_{\mathcal{J},\mathcal{J}} = \mathrm{diag}(s_{n-2^j+1}, s_{n-2^j}, \ldots, s_{n-2^{j+1}+2})$, hence s_{n-2^j+1} is the largest invariant factor of $G^{(0)}_{\mathcal{J},\mathcal{J}}$. In the next iteration, $G^{(1)}_{\mathcal{J},\mathcal{J}}$ is the Hermite basis of

$$\left[\begin{array}{c} G^{(0)}_{\mathcal{J},\mathcal{J}} \\ \hline G^{(0)}_{\mathcal{J},\mathcal{I}} \end{array}\right].$$

By Lemma 4.2, s_{n-2^j+1} is a multiple of the largest invariant factor of $G^{(1)}_{\mathcal{J},\mathcal{J}}$. Using similar arguments, s_{n-2^j+1} is a multiple of the largest invariant factor of $G^{(*)}_{\mathcal{J},\mathcal{J}}$.

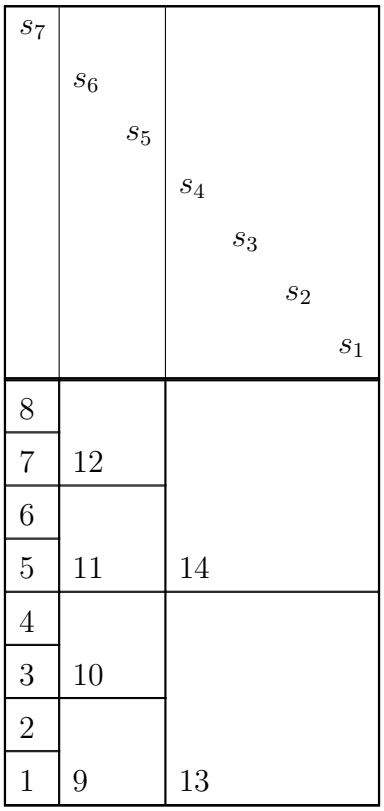


Figure 4.2: Blocking to compute the diagonal entries of the Hermite form.

Each iteration of Phase A costs $O(2^{j\omega}\,\mathsf{B}(\deg s_{n-2^j+1}))$ operations from K. For every $j = 0, 1, \cdots, t$, there are $O(n/2^j)$ iterations. We shall now use the following fact from Theorem 1.4

$$\deg s_k \leq nd/(n-k+1)$$

Thus there exists $c_2 > 0$, such that the total cost of all Block Eliminate steps is bounded by

$$c_2 \sum_{j=0}^{t} 2^{j\omega}\,\mathsf{B}(nd/2^j) \times n/2^j \leq c_2 \sum_{j=0}^{t} 2^{j\omega}\,\mathsf{B}(d)\mathsf{B}(n/2^j) \times n/2^j,$$

using the superlinearity of B (1.9). We now use the fast multiplication assumption (1.6) along with the bound on B (1.8) to get $\mathsf{B}(t) \in O((\mathsf{MM}(t)/t)\log t)$ and hence there exists a $c_3 > 0$:

$$c_2 \sum_{j=0}^{t} (2^{j\omega}\,\mathsf{B}(d)(n/2^j)^{\omega} \times \log(n/2^j)) \leq c_3(n^{\omega}\,\mathsf{B}(d) \sum_{j=0}^{t} \log(n/2^j)).$$

`BlockDiagonalHermite`(V, S)
Input: $V \in \mathsf{K}[x]^{n\times n}$, diagonal matrix $S \in \mathsf{K}[x]^{n\times n}$ is the Smith form of nonsingular $A \in \mathsf{K}[x]^{n\times n}$, with $d = \deg A$.
Output: $D \in \mathbb{Z}^{1\times n}$, the vector containing the degrees of the trailing n diagonal entries of the Hermite form of $\left[\begin{array}{c|c} S & \\ \hline V & I \end{array}\right]$

Initialize $D := [1, 1, \ldots, 1]$;
P is a $n \times n$ permutation matrix with ones on the anti diagonal;
$G = \left[\begin{array}{c|c} P & \\ \hline & I \end{array}\right] \left[\begin{array}{c} S \\ \hline V \end{array}\right] P$;

for j **from** 0 **to** t **do**
$\mathcal{J} := (2^j, \ldots, 2^{j+1} - 1)$;
  for i **from** $2n + 1$ **by** 2^j **to** $n + 2^j$ **do**
    $\mathcal{I} := (i - 2^j + 1, \ldots, i)$;
    A:[Block Eliminate]
    $\overbrace{\left[\begin{array}{c|c} H_1 & H_2 \\ \hline & H_4 \end{array}\right]}^{H} := \text{Hermite}\overbrace{\left(\left[\begin{array}{c|c} G_{\mathcal{J},\mathcal{J}} & \\ \hline G_{\mathcal{I},\mathcal{J}} & I \end{array}\right]\right)}^{W}$; /*$H = \phi^{-1}_{xs_{n-2^j+1}}(\text{Howell}(\phi_{xs_{n-2^j+1}}(W)))$*/
    $\left[\begin{array}{c} G_{\mathcal{J},\mathcal{J}} \\ \hline G_{\mathcal{I},\mathcal{J}} \end{array}\right] = \left[\begin{array}{c} H_1 \\ \hline \end{array}\right]$;
    B:[Apply Block Transformation]
    $U := H \overbrace{\left[\begin{array}{c|c} G^{-1}_{\mathcal{J},\mathcal{J}} & \\ \hline -G_{\mathcal{I},\mathcal{J}}G^{-1}_{\mathcal{I},\mathcal{I}} & I \end{array}\right]}^{W^{-1}}$;
    for k **from** $j + 1$ **to** t **do**
      $\mathcal{K} = (2^k, 2^k + 1, \ldots, 2^{k+1} - 1)$;
      $U := \text{Rem}(U, s_{n-2^k-1})$;
      $\left[\begin{array}{c} G_{\mathcal{J},\mathcal{K}} \\ G_{\mathcal{I},\mathcal{K}} \end{array}\right] := \text{Rem}\left(U \left[\begin{array}{c} G_{\mathcal{J},\mathcal{K}} \\ G_{\mathcal{I},\mathcal{K}} \end{array}\right], s_{n-k+1}\right)$;
    od
    C:[Update Block Diagonal]
    for every index k in $\mathcal{I}$ **do**
      $D[k] := D[k] \times H_4[k - i + 2^j]$;
    od
  od
od

return $[\deg D[1], \deg D[2], \ldots, \deg D[n]]$

Figure 4.3: Algorithm `BlockDiagonalHermite`

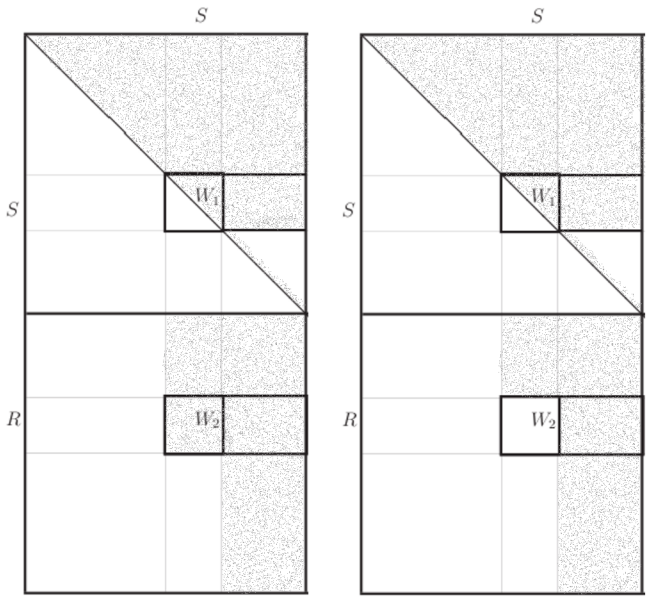


Figure 4.4: Shape of the work matrix after an iteration of Algorithm `BlockDiagonalHermite` in Figure 4.3.

Thus the cost of all Block Eliminate steps is $O(n^\omega(\log n)^2\,\mathsf{B}(d))$.

Now let us account for the cost of Phase B. The following lemma shows that the cost of all Apply Block Transformations is bounded by $O(n^\omega(\log n)^2\,\mathsf{M}(d))$.

Lemma 4.10. *The cost of all iterations of Phase B in* `BlockDiagonalHermite` *is bounded by* $O(n^\omega(\log n)^2\,\mathsf{M}(d))$. *Assuming that that* $\mathsf{M}(t) \in O(\mathsf{MM}(t)/t)$.

Proof. Note that $G_{\mathcal{J},\mathcal{J}}$ is in Hermite form. Thus $G^{-1}_{\mathcal{J},\mathcal{J}}$ is a proper matrix fraction. Hence $\deg U \leq \deg H \leq \deg(s_{n-2^j+1})$. Therefore, there exists a constant $c_4 > 0$, such that each iteration of Phase B costs

$$c_4(MM(2^j, \deg s_{n-2^j+1}) + \sum_{k=j+1}^{t} \mathsf{MM}(2^j, 2^j, 2^k, \deg s_{n-2^k-1})) = c_4(\sum_{k=j}^{t} \mathsf{MM}(2^j, 2^j, 2^k, \deg s_{n-2^k-1}))$$

operations. Also for every j there will be $O(n/2^j)$ iterations. Using the properties of MM; (1.5) and (1.4), we can state that there exist constants $c_5, c_6 > 0$ such that the total cost of all Apply Block Transformation steps is:

$$
\begin{aligned}
c_5 \left(\sum_{j=0}^{t} n/2^j \sum_{k=j}^{t} \mathsf{MM}(2^j, 2^j, 2^k, \deg s_{n-2^k-1}) \right) &= c_6 \left(\sum_{j=0}^{t} n/2^j \sum_{k=j}^{t} 2^{j\omega}\, 2^{k-j} \mathsf{M}(\deg s_{n-2^k-1}) \right) \\
&= c_6 \left(n \sum_{j=0}^{t} 2^{j(\omega-2)} \sum_{k=j}^{t} 2^k \mathsf{M}(\deg s_{n-2^k-1}) \right) \\
&= c_6 \left(n \sum_{k=1}^{t} 2^k \mathsf{M}(\deg s_{n-2^k-1}) \sum_{j=0}^{k} 2^{j(\omega-2)} \right) \\
&\leq 2c_6 \left(n \sum_{k=1}^{t} 2^k \mathsf{M}(\deg s_{n-2^k-1}) k 2^{k(\omega-2)} \right) \\
&= 2c_6 \left(n \sum_{k=1}^{t} k 2^{k(\omega-1)} \mathsf{M}(\deg s_{n-2^k-1}) \right) \\
&\leq 2c_6 \left(n \sum_{k=1}^{t} k 2^{k(\omega-1)} \mathsf{M}(nd/2^k) \right) \qquad (4.10) \\
&\leq 2c_6 \left(n \sum_{k=1}^{t} k 2^{k(\omega-1)} \mathsf{M}(n/2^k) \mathsf{M}(d) \right).
\end{aligned}
$$

Note that in (4.10), we use the bound on the degree of Smith invariants in Theorem 1.4. Now we use the assumption that $\mathsf{M}(t) \in O(\mathsf{MM}(t)/t)$. Thus, there exists a $c_6 > 0$ such that the cost of all Apply Block Transformation steps is bounded by:

$$
\begin{aligned}
2c_5 \left(\sum_{k=1}^{t} k 2^{k(\omega)} n/2^k \mathsf{M}(n/2^k) \mathsf{M}(d) \right) &\leq c_6 \left(\sum_{k=1}^{t} k n^\omega\, \mathsf{M}(d) \right) \\
&\leq c_6 (n^\omega (\log n)^2\, \mathsf{M}(d)).
\end{aligned}
$$

□

The cost of Update Block Diagonal steps remains the same as in the iterative case, that is, $O(n\mathsf{B}(nd))$. Thus, the total cost of finding the diagonal entries of the Hermite form of A using matrix multiplication, given S and V such that $A = VS^{-1}*$, is $O(n^\omega (\log n)^2\, \mathsf{B}(d))$.

We can now state the following result.

Theorem 4.11. *Algorithm* `BlockDiagonalHermite` *in Figure 4.3 is correct. The algorithm uses* $O(n^\omega (\log n)^2\, \mathsf{B}(d))$ *field operations from* K. *This cost estimate assumes that* $\mathsf{M}(t) \in \mathsf{MM}(t)/t$.

Chapter 5

Diagonal to Hermite Form

In this chapter we will recover the Hermite form of a nonsingular matrix $A \in \mathsf{K}[x]^{n\times n}$, using the information about the Smith form of A, the reduced Smith decomposition of A and the degrees of the diagonal entries of the Hermite form. Section 5.1 details a way to recover the Hermite form from a genset for a sub lattice of A. Section 5.2 gives a way to recover the Hermite form from the kernel basis of a matrix containing the Smith form of A and its reduced Smith transform. Finally Section 5.3 improves on Section 5.2 to compute a genset for a sub lattice of A in cost of the order of the matrix multiplication of polynomial matrices of the same dimension and degree bounds.

We begin by defining some notation. Let $\mathbf{e} = (e_1, \ldots, e_n)$ be a tuple of integers and $u = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix} \in \mathsf{K}[x]^{1\times n}$. Following [2], the $\mathbf{e}$-degree of u is equal to $\min_i \deg u_i - e_i$. We define $\mathcal{L}_{\mathbf{e}}(A)$ to be the set of row vectors of $\mathcal{L}(A)$ that have nonpositive $\mathbf{e}$-degree, that is, those vectors u that satisfy $\deg u_i \leq e_i$, $1 \leq i \leq n$.

Definition 5.1. *Let $L \in \mathsf{K}[x]^{*\times n}$ and $\mathbf{e} = (e_1, \ldots, e_n)$ be a tuple of degree constraints. A matrix $G \in \mathsf{K}[x]^{*\times n}$ is a* genset *of type $\mathbf{e} = (e_1, \ldots, e_n)$ for $\mathcal{L}_{\mathbf{e}}(L)$ if*

- *every row of G has nonpositive $\mathbf{e}$-degree,*
- *$\mathcal{L}_{\mathbf{e}}(G) = \mathcal{L}_{\mathbf{e}}(L)$.*

Note that for some tuples $\mathbf{e}$ we may have $\mathcal{L}(\mathcal{L}_{\mathbf{e}}(A)) \subset \mathcal{L}(A)$. In other words, there may not exist a basis for the lattice $\mathcal{L}(A)$ for which every row in the a basis has degree bounded by $\mathbf{e}$. An obvious example is when $\mathbf{e} = (-1, \ldots, -1)$, in which case $\mathcal{L}(\mathcal{L}_{\mathbf{e}}(A))$ has dimension zero.

5.1 From genset to Hermite form

Let $\mathbf{d} = (d_1, \ldots, d_n)$ be the degrees of the diagonal entries of the Hermite form H of a nonsingular $A \in \mathsf{K}[x]^{n\times n}$. Because H is a basis for $\mathcal{L}(A)$, and each row of H has nonpositive $\mathbf{d}$-degree, we have the following result.

Lemma 5.2. $\mathcal{L}(\mathcal{L}_{\mathbf{d}}(A)) = \mathcal{L}(A)$.

The following lemma shows how to recover H from a genset $\bar{H}$ of type $\mathbf{d}$ for $\mathcal{L}_{\mathbf{d}}(A)$. The lemma follows as a corollary of Lemma 5.2.

Lemma 5.3. *Suppose $\bar{H} \in \mathsf{K}[x]^{m\times n}$ is a genset of type $\mathbf{d}$ for $\mathcal{L}_{\mathbf{d}}(A)$ with $m \geq n$ and H is the Hermite form of A. Let $L \in \mathsf{K}^{m\times n} : \mathrm{Col}(L, j) = \mathrm{Coeff}(\mathrm{Col}(\bar{H}, j), x^{d_j})$, $1 \leq j \leq n$. If $U \in \mathsf{K}^{m\times m}$ is a nonsingular matrix such that UL is in reduced row echelon form, then $U\bar{H}$ will have principal $n \times n$ submatrix equal to H, and last $m - n$ rows zero.*

Proof. Suppose $U\bar{H}$ did not have principal $n \times n$ submatrix equal to H. Let the i, j - th entry of $U\bar{H}$ be $a \neq 0$ for some i, j such that $n \geq i > j$. Since UL is in row reduced form, the $\deg a < d_j$. This is a contradiction as we know that the degree of the j-th diagonal entry of the Hermite form is d_j, and from among all rows of $\mathcal{L}(A)$ which have first $i - 1$ entries zero and entry i nonzero, the i'th row of the Hermite form has i'th entry of minimal degree, $1 \leq i \leq n$.

Since the i, j entries of $U\bar{H}$ are zero for all $n \geq i > j$ and the degrees of diagonal entries are same as that of H, by uniqueness of the Hermite form the principal $n \times n$ submatrix of $U\bar{H}$ is H. □

Example 5.4. *Let $\mathsf{K} = \mathbb{Z}/(7)$, and consider the following Hermite form $H \in \mathsf{K}[x]^{3\times 3}$, together with a genset $\bar{H} \in \mathsf{K}[x]^{5\times 3}$ of type $(1, 3, 2)$ for $\mathcal{L}(H)$:*

$$H = \begin{bmatrix} x & x^2+1 & x+2 \\ & x^3+2x^2 & x+3 \\ & & x^2+2 \end{bmatrix},$$

$$\bar{H} = \begin{bmatrix} 4x & 6x^3+2x^2+4 & 6x^2+3x+3 \\ x & 4x^3+2x^2+1 & 5x^2+5x+3 \\ x & 2x^3+5x^2+1 & 3x^2+3x \\ 3x & 5x^3+6x^2+3 & 4x^2+x+1 \\ 2x & 2x^2+2 & 4x^2+2x+5 \end{bmatrix}.$$

The following shows the leading coefficient matrix L of $\bar{H}$, together with a nonsingular matrix $U \in \mathsf{K}^{5\times 5}$ that transforms L to reduced row echelon form, which due to Lemma 5.3

will necessarily have principle 3×3 *submatrix equal to* I_3.

$$\begin{matrix} & & U & & \\ \end{matrix} \begin{matrix} & & L & \\ \end{matrix}$$
$$\begin{bmatrix} 2 & 1 & 6 & 0 & 0 \\ 2 & 6 & 0 & 0 & 0 \\ 5 & 5 & 3 & 0 & 0 \\ 6 & 3 & 5 & 1 & 0 \\ 2 & 3 & 2 & 0 & 4 \end{bmatrix} \begin{bmatrix} 4 & 6 & 6 \\ 1 & 4 & 5 \\ 1 & 2 & 3 \\ 3 & 5 & 4 \\ 2 & 0 & 4 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \\ & & \\ & & \end{bmatrix}$$

$U\bar{H}$ *is equal to the Hermite form* H *augmented with two zero rows.*

5.2 Hermite form via kernel basis

The quantities defined in this subsection will be used in the remaining subsections. Let $A \in \mathsf{K}[x]^{n \times n}$ be nonsingular, with the following quantities precomputed:

- The Smith form $S = \text{Diag}(s_1, \ldots, s_n) \in \mathsf{K}[x]^{n \times n}$ of A.
- A matrix $V \in \mathsf{K}[x]^{n \times n}$ such that $u \in \mathcal{L}(A)$ if and only if uVS^{-1} is over $\mathsf{K}[x]$. The i'th column of V has entries of degree less than $\deg s_i$.
- The degrees $\mathbf{d} = (d_1, \ldots, d_n)$ of the diagonal entries of the Hermite form H of A.

Let $R := -HVS^{-1} \in \mathsf{K}[x]^{n \times n}$. Then

$$\left[\begin{array}{c|c} R & H \end{array}\right] \left[\frac{S}{V}\right] = 0. \tag{5.1}$$

Since column i of V has degree strictly less than $\deg s_i$, we have $\deg R \leq D$ where $D = \max_i d_i - 1$. Let $\mathbf{D} = (D, \ldots, D)$, of length n. The matrix $\left[\begin{array}{c|c} R & H \end{array}\right]$ is a basis (with all rows of nonpositive $(\mathbf{D}, \mathbf{d})$-degree) for the left kernel of $\left[\begin{array}{c|c} S & V^T \end{array}\right]^T$. In fact, by Lemma 5.3, to recover H it will be sufficient to compute a genset $\left[\begin{array}{c|c} \bar{R} & \bar{H} \end{array}\right]$ of type $(\mathbf{D}, \mathbf{d})$ for $\mathcal{L}_{(\mathbf{D},\mathbf{d})}(\left[\begin{array}{c|c} R & H \end{array}\right])$. The next subsection computes such a genset using fast minimal approximant basis computation.

We remark that the transformation of a canonical form computation to that of a kernel computation is used in [2, 3]. In particular, note that

$$\left[\begin{array}{c|c} U & H \end{array}\right] \left[\frac{A}{I_n}\right] = 0. \tag{5.2}$$

The setup in (5.2) requires no precomputation, and is useful if the unimodular transformation U to achieve the Hermite form is also required. What is important in our approach shown in (5.1) is the shape of the input problem: we will exploit the fact that S is diagonal, with sum of column degrees in both S and V bounded by nd.

5.3 Hermite via minimal approximant basis

Let $G \in \mathsf{K}[x]^{n\times m}$ and $\mathbf{e}$ be a tuple of nonnegative integers. The entries of $\mathbf{e}$ may be considered to be degree constraints. Recall that an order N minimal approximant basis (or σ-basis [1]) of type $\mathbf{e}$ for G is a nonsingular and row reduced matrix $M \in \mathsf{K}[x]^{n\times n}$ such that $MG \equiv 0 \bmod x^N$. The minimality condition means that the rows of M have $\mathbf{e}$-degrees as small as possible.

Lemma 5.5. *Let $M \in \mathsf{K}[x]^{2n\times 2n}$ be an order $N = D + \deg s_n + 1$ minimal approximant basis of type $(\mathbf{D}, \mathbf{d})$ for $\left[\begin{array}{c|c} S & V^T \end{array}\right]^T$. The submatrix of rows of M that have nonpositive $(\mathbf{D}, \mathbf{d})$-degree comprise a basis for $\mathcal{L}(\left[\begin{array}{c|c} R & H \end{array}\right])$.*

Proof. Let $v \in \mathsf{K}[x]^{1\times 2n}$ have nonpositive $(\mathbf{D}, \mathbf{d})$-degree. The order N is high enough that $v \left[\begin{array}{c|c} S & V^T \end{array}\right]^T = 0$ if and only if $v \left[\begin{array}{c|c} S & V^T \end{array}\right]^T \equiv 0 \bmod x^N$. □

Example 5.6. *Consider the matrix H from Example 5.4. The degrees of the diagonal entries of H are $(1, 3, 2)$ and thus $D = 2$. The Smith form of H is* $\operatorname{diag}(1, 1, x^6 + 2x^5 + 2x^4 + 4x^3)$. *Since the first two invariant factors are trivial, we can restrict S to its last entry and V to its last column as the input:*

$$\left[\frac{S}{V}\right] = \left[\begin{array}{c} x^6 + 2x^5 + 2x^4 + 4x^3 \\ \hline 5x^5 + x^3 + 6x^2 + 6 \\ 6x^5 + 3x^4 + 3x^3 + x \\ 4x^5 + 2x^4 + 2x^3 \end{array}\right]. \tag{5.3}$$

The following shows an order 9 minimal approximant basis M of type $(2, 1, 3, 2)$ for $\left[\begin{array}{c|c} S & V^T \end{array}\right]^T$, rows permuted to be in nondecreasing $(\mathbf{D}, \mathbf{d})$-degree.

$$M = \left[\begin{array}{ccc|ccc} & 3x + 6 & & & & x^2 + 2 \\ & x & & x & x^2 + 1 & x + 2 \\ & x^2 + 4x + 5 & & & x^3 + 2x^2 & x + 3 \\ \hline x^4 + 5x^3 + 2x^2 + x + 4 & & & 0 & 5x^2 & x \end{array}\right]$$

Exactly the first $n = 3$ rows have nonpositive $(\mathbf{D}, \mathbf{d})$-degree. For this example, the northeast block of M is the Hermite form of A up to a row permutation. In general, the northeast block will be a genset of full row rank for $\mathcal{L}_{\mathbf{d}}(H)$.

Using directly the approach of Lemma 5.5 to recover H is too expensive because the required order $N = D + \deg s_n + 1$ of the minimal approximant basis computation is too high. Indeed, we may have $N \in \Omega(nd)$. The reduction of order technique in [29, Section 2] can be used to reduce the order down to one more than times the maximum of the degree

constraints in $(\mathbf{D}, \mathbf{d})$. Unfortunately, the largest entry in $\mathbf{d}$ and $\mathbf{D}$ may be $\Omega(nd)$. Before applying the reduction of order technique we apply the partial linearization technique from [29, Section 3] to transform to a new minimal approximant basis problem of type $(\mathbf{D}, \mathbf{d}_1)$, with all entries of $\mathbf{d}_1$ bounded by d.

We need to recall some notation from [29]. The norm of a tuple of degree constraints $\mathbf{d}$ is defined to be $\|\mathbf{d}\| = (d_1 + 1) + \cdots + (d_n + 1)$. For $b \geq 0$, let ϕ_b be the function which maps a single degree bound d_i to a sequence of degree bounds, all element of the sequence equal to b except for possibly the last, and such that $\|(d_i)\| = d_i + 1 = \|(\phi_b(d_i))\|$. Let $\text{len}(\phi_b(d_i))$ denote the length of the sequence. For example, we have $\phi_3(10) = 3, 3, 2$ with $\text{len}(\phi_3(10)) = 3$, while $\phi_2(11) = 2, 2, 2, 2$ and $\text{len}(\phi_2(11)) = 4$. Computing a genset of type $(\mathbf{D}, \mathbf{d})$ for $\mathcal{L}_{(\mathbf{D},\mathbf{d})}(\left[\begin{array}{c|c} R & H \end{array}\right])$ can be reduced to computing an order N genset of type $\mathbf{d}_1 = (\phi_b(d_1), \ldots, \phi_b(d_n))$. Corresponding to $\mathbf{d}_1$ define the following $\bar{n} \times n$ expansion/compression matrix

$$B := \left[\begin{array}{c|c|c} \begin{matrix} 1 \\ x^{b+1} \\ \vdots \\ x^{(b+1)\text{len}(\phi_b(d_1))-1} \end{matrix} & & \\ \hline & \begin{matrix} 1 \\ x^{b+1} \\ \vdots \\ x^{(b+1)(\text{len}(\phi_b(d_2))-1)} \end{matrix} & \\ \hline & & \ddots \end{array}\right],$$

where $\bar{n} = \sum_i^n \text{len}(\phi_b(d_i)) = \sum_i^n \lceil (d_i + 1)/(b + 1) \rceil$.

Lemma 5.7. *Let $b \geq 0$ and define $e_i = \lceil (d_i + 1)/(b + 1) \rceil$, $1 \leq i \leq n$. Let M_1 be an order $N = D + \deg s_n + 1$ minimal approximant basis of type $(\mathbf{D}, \mathbf{d}_1)$ for $\left[\begin{array}{c|c} S & (BV)^T \end{array}\right]^T$, where $\mathbf{d}_1 = (\phi_b(d_1), \ldots, \phi_b(d_n))$. If $\left[\begin{array}{c|c} \bar{R}_1 & \bar{H}_1 \end{array}\right]$ is the subset of rows of M_1 which have degree bounded by $(\mathbf{D}, \mathbf{d}_1)$, then $\left[\begin{array}{c|c} \bar{R}_1 & \bar{H}_1 B \end{array}\right]$ is a genset of type $(\mathbf{D}, \mathbf{d})$ for $\mathcal{L}_{(\mathbf{D},\mathbf{d})}(H)$.*

Furthermore, with the choice $b = d$ the row dimension $\bar{n}$ of BV will satisfy $\bar{n} \in O(n)$.

Example 5.8. *The problem in Example 5.6 was to compute a minimal approximant of type $(\mathbf{D}, \mathbf{d}) = (2, 1, 3, 2)$ for the 4×1 input matrix shown in (5.3). Consider setting the linearization parameter b in Lemma 5.7 as $b = 1$. The expanded problem BV is*

$$\overset{B}{\left[\begin{array}{c|c|c} 1 & & \\ \hline & 1 & \\ & x^2 & \\ \hline & & 1 \\ & & x^2 \end{array}\right]} V = \overset{BV}{\left[\begin{array}{c} 5x^5 + x^3 + 6x^2 + 6 \\ \hline 6x^5 + 3x^4 + 3x^3 + x \\ 6x^7 + 3x^6 + 3x^5 + x^3 \\ \hline 4x^5 + 2x^4 + 2x^3 \\ 4x^7 + 2x^6 + 2x^5 \end{array}\right]}. \tag{5.4}$$

The degree constraints for the expanded problem are

$$\begin{aligned}(\mathbf{D}, \mathbf{d}_1) &= (2, \phi_1(1), \phi_1(3), \phi_1(2)) \\ &= (2, 1, 1, 1, 1, 0).\end{aligned}$$

The following shows an order 9 minimal approximant basis of type $(2,1,1,1,1,0)$ *for* $\left[\begin{array}{c|c} S & (BV)^T \end{array}\right]^T$ *.*

$$M = \left[\begin{array}{c|ccccc} 3x+6 & 0 & 0 & 0 & 2 & 1 \\ x^2+4x+5 & 0 & 0 & x+2 & x+3 & 0 \\ x & x & 1 & 1 & x+2 & 0 \\ \hline 3x+6 & 0 & 0 & 0 & x^2+2 & 0 \\ 0 & 0 & x^2 & 6 & 0 & 0 \\ x^4+5x^3+2x^2+x+4 & 0 & 0 & 5 & x & 0 \end{array}\right]$$

The first 3 *rows of* M *have nonpositive* $(\mathbf{D}, \mathbf{d}_1)$*-degree. Applying the compression matrix to the northwest block of* M *gives a genset* $\bar{H}$ *of type* $\mathbf{d}$ *for* $\mathcal{L}_{\mathbf{d}}(H)$*:*

$$\left[\begin{array}{ccccc} 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & x+2 & x+3 & 0 \\ x & 1 & 1 & x+2 & 0 \end{array}\right] B = \begin{array}{c} \bar{H} \\ \left[\begin{array}{ccc} 0 & 0 & x^2+2 \\ 0 & x^3+2x^2 & x+3 \\ x & x^2+1 & x+2 \end{array}\right] \end{array}.$$

Note that in this example $\bar{H}$ *has full row rank. We remark that, in general, the genset produced using this expansion/compression technique may have linearly dependent rows.*

At this point, we have reduced the problem of computing H to that of computing the rows $\left[\begin{array}{c|c} \bar{R}_1 & \bar{H}_1 \end{array}\right]$ of nonpositive $(\mathbf{D}, \mathbf{d}_1)$-degree in an order $N = D + \deg s_1 + 1$ minimal approximant basis of type $(\mathbf{D}, \mathbf{d}_1)$, namely

$$\left[\begin{array}{c|c} \bar{R}_1 & \bar{H}_1 \end{array}\right] \left[\begin{array}{c} S \\ \hline BV \end{array}\right] = 0 \bmod x^N.$$

The degree constraints $\mathbf{D} = (D, \ldots, D)$ corresponding the columns of $\bar{R}_1$ may still be too large in general, since $D = \max_i d_i - 1 \in \Omega(nd)$ in the worst case. The key idea now is that $\bar{R}_1$ is not required. Let C be a matrix such that $BV - CS$ has each column of degree bounded by s_i, and consider the transformed input:

$$\left[\begin{array}{c|c} I_n & \\ \hline -C & I \end{array}\right] \left[\begin{array}{c} S \\ \hline BV \end{array}\right] = \left[\begin{array}{c} S \\ \hline E \end{array}\right]. \tag{5.5}$$

Note that each column in E has degree strictly less than the corresponding diagonal entry in S.

Lemma 5.9. *Let* $\mathbf{D}_1 = (d-1, \ldots, d-1)$*, of length* n*. Let* M_2 *be an order* $N = D + \deg s_n + 1$ *minimal approximant basis of type* $(\mathbf{D}_1, \mathbf{d}_1)$ *for* $\left[\begin{array}{c|c} S & E^T \end{array}\right]^T$*. Let* $\left[\begin{array}{c|c} \bar{R}_2 & \bar{H}_2 \end{array}\right]$ *be the submatrix of* M_2 *comprised of rows that have nonpositive* $(\mathbf{D}_1, \mathbf{d}_1)$*-degree. Then* $\bar{H}_2 B$ *is a genset of type* $\mathbf{d}$ *for* $\mathcal{L}_{\mathbf{d}}(H)$*.*

Proof. The order N is large enough to ensure that

$$\left[\begin{array}{c|c} \bar{R}_2 & \bar{H}_2 \end{array}\right] \left[\begin{array}{c|c} S & E^T \end{array}\right]^T = 0,$$

and (5.5) gives that $\left[\begin{array}{c|c} \bar{R}_2 - \bar{H}_2 C & \bar{H}_2 \end{array}\right] \left[\begin{array}{c|c} S & (BV)^T \end{array}\right]^T = 0$, which implies that

$$\left[\begin{array}{c|c} \bar{R}_2 - \bar{H}_2 C & \bar{H}_2 B \end{array}\right] \left[\frac{S}{V}\right]^T = 0,$$

with all rows in $\bar{H}_2 B$ of nonpositive $\mathbf{d}$-degree. But since V has degrees of entries bounded by the corresponding diagonal entry of S, each row of $\bar{R}_2 - \bar{H}_2$ has nonpositive $\mathbf{D}$ degree. We conclude that $\left[\begin{array}{c|c} \bar{R}_2 - \bar{H}_2 C & \bar{H}_2 \end{array}\right] \subseteq \mathcal{L}_{(\mathbf{D},\mathbf{d})}(\left[\begin{array}{c|c} R & H \end{array}\right])$. The other direction is similar. □

Provided we have chosen the linearization parameter b in Lemma 5.7 to satisfy $b \in \Theta(d)$ (e.g., $b = d$ will suffice), the final minimal approximant problem in Lemma 5.9 will have dimension $O(n) \times n$. Note that entries of the compression/expansion matrix B are all powers of x. Thus, the only computation (in terms of field operations) required to construct the input problem in Lemma 5.9 is to construct E from BV by reducing entries in each column i modulo the diagonal entry in the same column of S, $1 \leq i \leq n$.

Example 5.10. *The problem in Example 5.8 was to compute a minimal approximant of type* $(\mathbf{D}, \mathbf{d}_1) = (2, 1, 1, 1, 1, 0)$ *for the partially linearized* 6×1 *input matrix* B *shown in (5.4). Reducing the last 5 entries modulo the the principal entry we obtain the new input*

$$\left[\frac{S}{E}\right] = \left[\begin{array}{c} \frac{x^6 + 2x^5 + 2x^4 + 4x^3}{} \\ \hline 5x^5 + x^3 + 6x^2 + 6 \\ 6x^5 + 3x^4 + 3x^3 + x \\ 2x^5 + x^4 + 2x^3 \\ \hline 4x^5 + 2x^4 + 2x^3 \\ 6x^5 + 3x^4 + 3x^3 \end{array}\right]. \tag{5.6}$$

The following shows the submatrix of an order $N = D + \deg s_n + 1 = 9$ *minimal approximant basis of type* $(\mathbf{D}_1, \mathbf{d}_1) = (0, 1, 1, 1, 1, 0)$ *for* $\left[\begin{array}{c|c} S & E^T \end{array}\right]^T$ *comprised of rows that have nonpositive* $(\mathbf{D}_1, \mathbf{d}_1)$*-degree:*

$$\left[\begin{array}{c|c} \bar{R}_2 & \bar{H}_2 \end{array}\right] = \left[\begin{array}{c|ccccc} 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & x & 1 & 2x+5 & 3x+1 & 0 \\ 1 & 0 & 0 & x+2 & x+3 & 0 \end{array}\right].$$

Applying the compression matrix B *to* $\bar{H}_2$ *yields a genset* $\bar{H}_2 B$ *of type* $\mathbf{d}$ *for* $\mathcal{L}_{\mathbf{d}}(H)$*.*

At this point (Lemma 5.9) we have reduced the problem of computing H to that of computing the rows $\left[\begin{array}{c|c} \bar{R}_2 & \bar{H}_2 \end{array}\right]$ of nonpositive $(\mathbf{D}_1, \mathbf{d}_1)$-degree of a minimal approximant basis of order $N = D + \deg s_n + 1$ for an input $\left[\begin{array}{c|c} S & E^T \end{array}\right]^T$. If the partial linearization parameter in Lemma 5.7 was chosen as $b = d$, then E has dimension $O(n) \times n$, and all degree constrains in $(\mathbf{D}_1, \mathbf{d}_1)$ are bounded by d. Since the sum of the column degrees in $\left[\begin{array}{c|c} S & E^T \end{array}\right]^T$ is bounded by nd, the reduction of order technique in [29, Section 2] can be used to transform to an equivalent problem of dimension $O(n) \times O(n)$ and order only $2d + 1$. We refer to [29] for details of the reduction of order technique, and only illustrate the technique here on our running example.

Example 5.11. *In Example 5.10 we computed an order 9 minimal approximant basis of type* $(\mathbf{D}_1, \mathbf{d}_1) = (0, 1, 1, 1, 1, 0)$ *for the* 6×1 *input* $F := \left[\begin{array}{c|c} S & E^T \end{array}\right]^T$ *shown in (5.6). Since the maximum degree constraint is* 1*, we can instead compute an order* $2 \cdot 1 + 1 = 3$ *minimal approximant basis* $\bar{M}$ *of type* $(\mathbf{D}_1, \mathbf{d}_1, d-1, d-1) = (0, 1, 1, 1, 1, 0, 0, 0)$ *for the following input:*

$$\bar{F} = \left[\begin{array}{c|c|c} F & \mathrm{Quo}(F, x^2) & \mathrm{Quo}(F, x^4) \\ \hline & 1 & \\ & & 1 \end{array}\right] \in \mathsf{K}[x]^{8\times 3}.$$

Indeed, the submatrix of $\bar{M}$ *comprised of rows that have nonpositive* $(\mathbf{D}_1, \mathbf{d}_1, 0, 0)$*-degree can be written as* $\left[\begin{array}{c|c} W & * \end{array}\right]$*, where* W *is the submatrix of an order* 9 *minimal approximant basis of type* $(\mathbf{D}_1, \mathbf{d}_1)$ *for* F*.*

Theorem 5.12. *Let* $A \in \mathsf{K}[x]^{n\times n}$ *be nonsingular of degree* d*. Assuming* $\#\mathsf{K} \geq 8n^3 d$*, there exists a Las Vegas probabilistic algorithm that computes the Hermite form* H *of* A *with probability greater than* $1/2$ *or returns fail, using* $O(n^\omega (log n)^2\, \mathsf{B}(d))$ *field operations from* K*.*

Proof. By Theorem 4.8, the Smith form S of A and corresponding V as described in Subsection 5.2 can be computed in a Las Vegas fashion in the allotted time. By Theorem 4.11, the degrees of the diagonal entries of H can be computed in the allotted time using Algorithm `BlockDiagonalHermite` in Figure 4.3. Construct column i of the block E of the input $\left[\begin{array}{c|c} S & E^T \end{array}\right]^T$ to the minimal approximant problem of Lemma 5.9 by reducing modulo s_i the entries in column i of BV, $1 \leq i \leq n$. Compute the rows of nonpositive degree in the minimal approximant indicate in Lemma 5.9 by first applying the reduction of order technique from [29, Section 2] to obtain a new problem of dimension $O(n) \times O(n)$ and order $2d + 1$, and then apply algorithm `PM-Basis` from [11] in time $O(n^\omega\, \mathsf{B}(d))$ operations from K. Finally, use the approach of Lemma 5.3 to recover the Hermite form from the genset for $\mathcal{L}_{\mathbf{d}}(H)$. □

Chapter 6

Conclusion

In this chapter we collate the results in the previous chapters to give an algorithm to compute the Hermite form of a nonsingular $A \in \mathsf{K}[x]^{n\times n}$. The algorithm is described in Figure 6.1. The central result of this thesis is the following.

Theorem 6.1. *Let $A \in \mathsf{K}[x]^{n\times n}$ be nonsingular of degree d. Algorithm* `Hermite`(A, n, d) *in Figure 6.1 is correct and it computes the Hermite form H of A with probability greater than $1/2$ or returns fail, using $O(n^\omega(\log n)^2\,\mathsf{B}(d))$ field operations from K.*

The preconditioning phase of Algorithm `Hermite` in Figure 6.1, makes sure that the preconditions of Algorithm `RST` in Figure 3.2 are satisfied. The deterministic row reduction algorithm from [12] takes $O(n^\omega(\log n)^2\,\mathsf{B}(d))$ basic operations from K. The probabilistic algorithm [11] can also be used here for row reduction and it costs $O(n^\omega(\log n)\mathsf{B}(d))$ basic operations from K. The Smith form of A can be computed using [28] in cost $O(n^\omega(\log n)^2\,\mathsf{B}(d))$ operations from K. It is a Las Vegas type algorithm which succeeds when the determinant of the input matrix has a non zero constant term. After the end of the preconditioning phase, we will work to compute the Hermite form of R, a row reduced form of A. Since, $R \equiv_L$, the Hermite form of R gives the Hermite form of A.

After A has been preconditioned into R', the computation of a reduced Smith transform phase computes a reduced Smith transform of R, using Algorithm `RST` in Figure 3.2 in cost $O(n^\omega(\log n)^2\,\mathsf{M}(d))$ operations from K (see Theorem 4.8). It is a Las Vegas type algorithm which succeeds with probability at least $1/2$ when the field K has at least $8n^3d$ elements. In case the field is small we can work over an algebraic extension field of K at an added cost. Algorithm `BlockDiagonalHermite` in Figure 4.3 is a deterministic algorithm, and costs $O(n^\omega(\log n)^2\,\mathsf{B}(d))$ operations from K (see Theorem 4.11).

Finally, in fast minimal approximant basis phase, we use the information computed in all the previous phases to compute the Hermite form of A. This phase consists of deterministic procedures and costs $O(n^\omega\,\mathsf{B}(d))$ operations from K (see Theorem 5.12).

`Hermite`(A, n, d)
Input: Nonsingular $A \in \mathsf{K}[x]^{n\times n}$, with $d = \deg A$.
Condition: $\#\mathsf{K} \geq 8n^3d$.
Output: $H \in \mathsf{K}[x]^{n\times n}$, the Hermite form of A or fail.

[Preconditioning]

$R :=$ Row reduced form of $A \in \mathsf{K}[x]^{n\times n}$; /* Using [12] */
$L, U \in \mathsf{K}^{n\times n}$ be unit lower and unit upper triangular matrices with non-zero non diagonal entries chosen uniformly at random from K;
$R' := LRU \in \mathsf{K}[x]^{n\times n}$;
$S :=$ SmithForm(A); /*Using [28]*/

[Compute a reduced Smith Transform]
$(U', V') :=$ `RST`$(R', S) \in \mathsf{K}[x]^{n\times n}$;
$V := UV' \in \mathsf{K}[x]^{n\times n}$;

[Compute the degree of the diagonal entries of the Hermite form]
d :=`BlockDiagonalHermite`(V, S);

[Fast minimal approximant basis]
Recover H by computing the minimal approximant basis of $\left[\frac{S}{V}\right]$ /* Chapter 5 */
return H;

Figure 6.1: Algorithm `Hermite Form Computation`

As expected the computation of the Hermite form of a nonsingular matrix is at least as hard, up to a constant factor, as multiplication of two matrices if the degree bounds in both cases are the same. The reduction of matrix multiplication to computation of the Hermite from was first shown by [11]. We present another reduction of matrix multiplication to the computation of the Hermite form. We show that we can compute the product of two matrices A and $B \in \mathsf{K}[x]^{n\times n}$ of degree bounded by d, by computing the Hermite form of a matrix whose dimensions are in $O(n)$. Consider the following matrix:

$$E := \left[\begin{array}{c|c} -I & C \\ \hline C & \end{array}\right].$$

Where C is a nonsingular square matrix in Hermite form with all diagonal entries of the

same degree. For $C \in \mathsf{K}[x]^{4\times 4}$, the matrix C and C^2 will look like the following:

$$C = \begin{bmatrix} x^d + * & (d) & (d) & (d) \\ & x^d + * & (d) & (d) \\ & & x^d + * & (d) \\ & & & x^d + * \end{bmatrix},$$

$$C^2 = \begin{bmatrix} x^{2d} + * & (2d) & (2d) & (2d) \\ & x^{2d} + * & (2d) & (2d) \\ & & x^{2d} + * & (2d) \\ & & & x^{2d} + * \end{bmatrix}.$$

Therefore, if C is in Hermite form with all diagonal entries of the same degree then C^2 will also be in Hermite form. The following will be the Hermite transformation for E:

$$\left[\begin{array}{c|c} I & \\ \hline C & I \end{array}\right] \left[\begin{array}{c|c} -I & C \\ \hline C & \end{array}\right] = \left[\begin{array}{c|c} -I & C \\ \hline & C^2 \end{array}\right].$$

Thus computing the Hermite form of the matrix E gives us the square of matrix C. Now let us define C as the following:

$$C = \left[\begin{array}{cc|cc} x^{d+1}I_n & A & 0 & \\ & x^{d+1}I_n & & B \\ \hline & & x^{d+1}I_n & \\ & & & x^{d+1}I_n \end{array}\right] \in \mathsf{K}[x]^{4n\times 4n}.$$

C is in Hermite form and all its diagonal entries are same. C^2 has the following shape:

$$C^2 = \left[\begin{array}{cc|cc} x^{2(d+1)}I_n & 2x^{d+1}A & 0 & AB \\ & x^{2(d+1)}I_n & & 2x^{d+1}B \\ \hline & & x^{2(d+1)}I_n & \\ & & & x^{2(d+1)}I_n \end{array}\right].$$

Thus we can multiply any two matrices $A, B \in \mathsf{K}[x]^{n\times n}$ with degree bounded by d by finding the Hermite form of a matrix $E \in \mathsf{K}[x]^{8n\times 8n}$ with degree bounded by $d+1$.

Algorithm `Hermite` in Figure 6.1 can be easily extended to find the Hermite form of a rectangular matrix of an arbitrary rank (see [26, Chapter 6]). Also, we can easily find a Hermite decomposition of $A = UH$, given the Hermite form H of A. The the degree of U is bounded by the degree of A, hence the operation $U = \mathrm{Rem}(AH^{-1}, X)$, can be carried out in cost $O(n^\omega, \mathsf{B}(d))$, where X is an polynomial of degree $d+1$ such that $X \perp \det A$.

The problem of derandomizing the computation of a Hermite decomposition is at least as hard as the problem of derandomizing the computation of a Smith decomposition. The

randomization in Algorithm `Hermite` in Figure 6.1 stems from the randomization needed to compute the Smith form and a reduced Smith transform which can be found by a reduced Smith decomposition of $s_n A^{-1}$.

The Hermite form of integer matrices has applications in integer programming and solutions to linear Diophantine equations. Unfortunately, we cannot directly use the Algorithm `Hermite` in Figure 6.1 to compute the Hermite form of integer matrices. The major hurdle to extend Algorithm `Hermite` in Figure 6.1 to integer matrices is that there is no equivalent of minimal approximant basis over $\mathbb{Z}$. Also, there is a problem of carry with integer computations. It would be possible to get the diagonal entries of the Hermite form of integer matrices using the techniques in this thesis but we would need a different approach to go from diagonal entries to the computation of the Hermite form.

Bibliography

[1] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix–type Padé approximants. *SIAM Journal on Matrix Analysis and Applications*, 15(3):804–823, 1994. 71

[2] B. Beckermann, G. Labahn, and G. Villard. Shifted normal forms of polynomial matrices. In S. Dooley, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '99*, pages 189—196. ACM Press, New York, 1999. 68, 70

[3] B. Beckermann, G. Labahn, and G. Villard. Normal forms for general polynomial matrices. *Journal of Symbolic Computation*, 41(6):708–737, 2006. 70

[4] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1996. 1

[5] P. D. Domich. Three new polynomially-time bounded Hermite normal form algorithms. Master's thesis, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1983. 1

[6] P. D. Domich. *Residual Methods for Computing Hermite and Smith Normal Forms*. PhD thesis, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1985. 1

[7] P. D. Domich. Residual Hermite normal form computations. *ACM Trans. Math. Software*, 15:275–286, 1989. 50

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

[9] W. Eberly, M. Giesbrecht, and G. Villard. Computing the determinant and Smith form of an integer matrix. In *Proc. 31st Ann. IEEE Symp. Foundations of Computer Science*, pages 675–685, 2000. 7

[10] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2nd edition, 2003. 3

[11] P. Giorgi, C.-P. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In R. Sendra, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '03*, pages 135–142. ACM Press, New York, 2003. 1, 2, 10, 56, 75, 76, 77

[12] 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*, October 2010. Festschrift for the 60th Birthday of Joachim von zur Gathen. Accepted for publication. 8, 39, 56, 76, 77

[13] S. Gupta and A. Storjohann. Computing hermite forms of polynomial matrices. In A. Leykin, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '11*, pages 155–162. ACM Press, New York, 2011. 1

[14] J. L. Hafner and K. S. McCurley. Asymptotically fast triangularization of matrices over rings. *SIAM Journal of Computing*, 20(6):1068–1083, December 1991. 1, 50

[15] J. A. Howell. Spans in the module $(\mathbb{Z}_m)^s$. *Linear and Multilinear Algebra*, 19:67—77, 1986. 51

[16] C. S. Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix. *SIAM Journal of Computing*, 18(4):658–669, 1989. 1, 50

[17] C. S. Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of infinite abelian groups and solving systems of linear diophantine equations. *SIAM Journal of Computing*, 18(4):670–678, 1989. 50

[18] C. P. Jeannerod and G. Villard. Essentially optimal computation of the inverse of generic polynomial matrices. *Journal of Complexity*, 21:72–86, 2005. 2

[19] T. Kailath. *Linear Systems*. Prentice Hall, Englewood Cliffs, N.J., 1980. 6, 7

[20] E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM Journal of Algebraic and Discrete Methods*, 8:683–690, 1987. 33, 38

[21] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Computational Complexity*, 13(3–4):91–130, 2004. 2

[22] S. E. Labhalla. *Complexité en temps polynomial : calcul d'une réduite d'Hermite, les différentes représentations des nombres réels*. Doctorat d'Etat, Université Cadi Ayyad, Faculté des Sciences Semlalia, Marrakech, 1991. 1

[23] D. Micciancio and B. Warinschi. A linear space algorithm for computing the Hermite normal form. In B. Mourrain, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '01*, pages 231—236. ACM Press, New York, 2001. 1

[24] T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *Journal of Symbolic Computation*, 35(4):377–401, 2003. 1

[25] A. Storjohann. Computation of Hermite and Smith normal forms of matrices. Master's thesis, Dept. of Computer Science, University of Waterloo, 1994. 1

[26] A. Storjohann. *Algorithms for Matrix Canonical Forms*. PhD thesis, Swiss Federal Institute of Technology, ETH–Zurich, 2000. 1, 54, 78

[27] A. Storjohann. High–order lifting. Extended Abstract. In T. Mora, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '02*, pages 246–254. ACM Press, New York, 2002. 1, 7, 82

[28] A. Storjohann. High–order lifting and integrality certification. *Journal of Symbolic Computation*, 36(3–4):613–648, 2003. Extended abstract in [27]. 1, 31, 32, 48, 76, 77

[29] A. Storjohann. Notes on computing minimal approximant bases. In W. Decker, M. Dewar, E. Kaltofen, and S. Watt, editors, *Challenges in Symbolic Computation Software*, number 06271 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. 10, 71, 72, 75

[30] A. Storjohann. On the complexity of inverting integer and polynomial matrices. *Computational Complexity*, 2010. Accepted for publication. 2, 8, 32, 47

[31] 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. 1

[32] A. Storjohann and T. Mulders. Fast algorithms for linear algebra modulo N. In G. Bilardi, G. F. Italiano, A. Pietracaprina, and G. Pucci, editors, *Algorithms — ESA '98*, LNCS 1461, pages 139–150. Springer Verlag, 1998. 54

[33] G. Villard. Computing Popov and Hermite forms of polynomial matrices. In Y. N. Lakshman, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '96*, pages 251–258. ACM Press, New York, 1996. 1

[34] U. Vollmer. A note on the Hermite basis computation of large integer matrices. In R. Sendra, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '03*, pages 255–257. ACM Press, New York, 2003. 1

[35] C. Wagner. *Normalformberechnung von Matrizen über euklidischen Ringen*. PhD thesis, Universität Karlsruhe, 1998. 1

[36] W. Zhou and G. Labahn. Efficient computation of order basis. In J. P. May, editor, *Proc. Int'l. Symp. on Symbolic and Algebraic Computation: ISSAC '09*, pages 375–384. ACM Press, New York, 2009. 2