Article published in Journal of Symbolic Computation (Volume 32, Issues 1-2, July 2001, Pages 71-99)

On efficient sparse integer matrix Smith normal form computations*

JEAN-GUILLAUME DUMAS[1], B. DAVID SAUNDERS[2] AND GILLES VILLARD[3]

[1]*Unité Informatique et Distribution, ENSIMAG - antenne de Montbonnot. ZIRST - 51, av. Jean Kuntzmann, 38330 Montbonnot Saint-Martin, France.*
Jean-Guillaume.Dumas@imag.fr, www-apache.imag.fr/~ jgdumas

[2]*Department of Computer and Information Sciences, University of Delaware, Newark, Delaware 19716.*
saunders@udel.edu, www.eecis.udel.edu/~ saunders

[3]*Laboratoire de l'Informatique du Parallélisme, ENSL - 46, allée d'Italie, F-69364 Lyon Cedex 07, France.*
Gilles.Villard@ens-lyon.fr, www.ens-lyon.fr/~ gvillard

Abstract

We present a new algorithm to compute the Integer Smith normal form of large sparse matrices. We reduce the computation of the Smith form to independent, and therefore parallel, computations modulo powers of word-size primes. Consequently, the algorithm does not suffer from coefficient growth. We have implemented several variants of this algorithm (Elimination and/or Black-Box techniques) since practical performance depends strongly on the memory available. Our method has proven useful in algebraic topology for the computation of the homology of some large simplicial complexes.

1. Introduction

In this article we study the computation of the integer Smith form of sparse matrices. The classical Smith form algorithm performs an elimination process with some gcd computations over the integers or modulo large primes [Rayward-Smith, 1979; Bachem and Kannan, 1979; Iliopoulos, 1989]. The best known complexities may be found in [Storjohann, 1996, 2000] for a deterministic algorithm or in [Eberly et al., 2000] for a probabilistic algorithm of the Monte Carlo type. For sparse matrices, one should expect to accelerate the solution by exploiting the sparsity. There exits few theoretical advances in this direction. The method

*Some of this work was done while the first author was visiting the University of Delaware in [Giesbrecht, 1996] is based on iterative methods and does substantially better for sparse matrices than above cited methods, but the approach is not very practical yet.

Our new probabilistic algorithm reduces the Smith form to computations modulo powers of small primes. Consequently the algorithm does not suffer from coefficient growth. Moreover, the modular computations are independent of each other, permitting an easy and very effective parallelization. Depending on some space/time tradeoff considerations, we may choose either iterative or direct methods at certain stages of our algorithm. Some salient features of our approach are that (1) we use the ovals of Cassini to often get a better determinant bound than Hadamard's and (2) we begin with the trailing coefficient (valence) of the minimal polynomial of the symmetrized but unpreconditioned matrix AA^t. The method is particularly effective when this polynomial is of low degree and hence fast to compute by Wiedemann's method. This has proven to be the case for many of the boundary map matrices of the simplicial complexes given to us by Volkmar Welker [Björner and Welker, 1999; Babson et al., 1999]. We report on experiments involving these matrices which arise in the computation of the homology of the complexes. Indeed, the reduced homology of a simplicial complex is equivalent information to the integer Smith form of its boundary maps [Munkres, 1994]. In the cases we studied, the boundary maps can be very large, e.g. around 10^5 rows and columns, and very sparse, e.g. 6 nonzero entries per row.

A preliminary version of this article appeared in [Dumas et al., 2000]. Here we give sharper estimates on the number of primes involved, give a complete probabilistic analysis of the Valence computation, include an analysis of the asymptotic space and time cost of the algorithm, and offer more experimental results.

We present an overview of our algorithm in §2. It works in three main steps: I/ Computation of the valence; II/ Construction – from the valence – of a set L of primes candidates for being factors of the entries of the Smith form; III/ Reconstruction of the Smith form from the local Smith forms at the primes in L. Step I/ is implemented in §3 where an integer minimal polynomial has to be computed. The main concern is to take advantage of some special properties of the boundary map matrices, to reduce the number of primes used in the proposed homomorphic scheme. Step II/ is studied in §4 where a precise characterization of the primes occurring in the valence and involved in the Smith form is given. This leads to a very small set L of primes candidates for the next step. The latter is detailed in §5 where we show how to compute the local Smith form at a given prime p. Once these are computed at all the primes of L the Smith form itself is easily derived. We also propose a specialized memory efficient algorithm to check the value of the largest entry of the Smith form.

This presentation of the algorithm is followed by asymptotic cost analyses in §6. Finally we report on experiments using matrices from homology in §7 and demonstrate how effective our approach can be.

2. Valence based Smith form algorithm

In this section we describe an algorithm for the computation of the Smith form of an integer matrix. This algorithm has proven effective on some of the boundary matrices discussed in this paper, though the worst case asymptotic complexity is not better than for Giesbrecht's algorithm [Giesbrecht, 1996, Theorem 2.5]. The method is particularly effective when the degree of the minimal polynomial of AA^t is small. We begin with some definitions:

Definitions 2.1:

- *The* **valence** *of a polynomial is its trailing nonzero coefficient. By extension, The* **characteristic valence** *of a matrix is the valence of its characteristic polynomial. The* **minimal valence** *or simply the* **valence** *of a matrix is the valence of its minimal polynomial.*
- *The* **valuation** *is the degree of the corresponding term. The* **characteristic** *and* **minimal valuations** *of a matrix are similarly defined.*
- *For $1 \leq i \leq min(m, n)$, the* **i-th determinantal divisor** *of a matrix A, $d_i(A)$, is the greatest common divisor of the $i \times i$ minors of A. Define $d_0(A) = 1$.*
- *For $1 \leq i \leq min(m, n)$, the* **i-th invariant factor** *of A is $s_i(A) = d_i(A)/d_{i-1}(A)$, or 0 if $d_i(A) = 0$. Let $s_0 = 1$.*
- *The* **Smith form** *of A is the diagonal matrix $S = diag(s_1(A), \ldots, s_{\min(m,n)}(A))$ of the same shape as A.*

It is well known that for $1 \leq i \leq \min(m, n)$, we have $d_{i-1}|d_i$, $s_{i-1}|s_i$, and that A is unimodularly equivalent to its Smith form.

Notations 2.1:

- *For a positive integer q, we denote by $\mathbb{Z}_q$ the quotient ring $\mathbb{Z}/q\mathbb{Z}$.*
- *The set of invertible elements in $\mathbb{Z}_q$ is denoted by ${\mathbb{Z}_q}^*$.*

Definitions 2.2:

- *For a positive integer q, we define* rank of A mod q, *to be the greatest i such that q does not divide the i-th invariant factor of A, and we denote this rank by $r_q = \text{rank}_q(A)$.*
- *The rank of A as an integer matrix will be denoted $r = \text{rank}(A)$.*

First we present an overview of the valence method. The individual steps can be done in many ways. Afterwards we will discuss the implementation details.

```
Algorithm: VSF [Valence-Smith-Form]
Input:   – a matrix $A \in \mathbb{Z}^{m\times n}$. A may be a "black box" meaning that the only
           requirement is that left and right matrix-vector products may be
           computed: $x \longrightarrow Ax$ for $x \in \mathbb{Z}^n$, $y \longrightarrow yA$ for $y^t \in \mathbb{Z}^m$.
Output:  – $S = \mathrm{diag}(s_1, \dots, s_{\min(m,n)})$, the integer Smith form of $A$.

(1)   [Valence computation]
      If $m < n$, let $B = AA^t$, otherwise let $B = A^tA$.
      Let $N = \min(m, n)$
      Compute the (minimal) valence $v$ of $B$.                    [See section 3 ]

(2)   [Integer factorization]
      Factor the valence $v$.
      Let $L$ be the list of primes which divide $v$.
      [If $v$ is hard to factor, see section 6 ]

(3)   [Rank and first estimate of Smith form.]
      Choose a prime $p$ not in $L$ (i.e. $p \nmid v$).
      Compute $r = \mathrm{rank}_p(A)$. [ This is the integer rank. The first $r$ invariant
      factors are nonzero and the last $N - r$ are 0's. ]
      Set $S = \mathrm{diag}(s_1, \dots, s_N)$, where $s_i = 1$ for $i \le r$ and $s_i = 0$ for $i > r$.

(4)   [Nontrivial invariant factors]
      For each prime $p \in L$
        Compute $S_p = \mathrm{diag}(s_{p,1}, \dots, s_{p,N})$, the Smith form of $A$ over the local
      ring $\mathbb{Z}^{(p)}$.                                           [See section 5 ]
        Set $S = SS_p$.         [That is, set $s_i = s_i s_{p,i}$ for those $s_{p,i}$ which are
      nontrivial powers of $p$. ]

(5)   [Return invariants]
      Return $S = \mathrm{diag}(s_1 \dots s_N) \in \mathbb{Z}^{m\times n}$.
```

In order to prove the correctness of the method we will need the following theorem.

Theorem 2.1: *Let A be a matrix in $\mathbb{Z}^{m\times n}$. Let $(s_1, \dots, s_r)$ be its nonzero invariant factors. If a prime $p \in \mathbb{Z}$ divides some nonzero s_i, then p^2 divides the characteristic valence of AA^t and p divides the minimal* valence(AA^t). *The same is true of the valences of A^tA as well.*

Proof: Let $B = A^tA$. The argument will apply equally well to $B = AA^t$. Let $M(x) = \mathrm{minpoly}(B)$ and let $v = \mathrm{valence}(M) = \mathrm{valence}(B)$. Let $C(x) = \mathrm{charpoly}(B)$, and let $F_1(x), \dots F_k(x)$ be the invariant factors of B. It is well known that these are monic integer polynomials with

$$F_1(x)|F_2(x)|\dots|F_k(x) = \text{minpoly}(B) \text{ and } C(x) = \prod F_i(x).$$

Let $v_c = \text{valence}(C) = \text{characteristic valence}(B)$. and let $v_i = \text{valence}(F_i)$. It follows easily from the nature of polynomial arithmetic that $v_1|v_2|\dots|v_k = v$, and $v_c = \prod v_i$, Hence all primes that occur in v_c occur in v. Thus the second part of the conclusion follows from the first and it suffices now to show any prime occurring in the Smith form of A occurs squared in v_c:

Notations 2.2: *As in [Kaltofen et al., 1990], we denote by S_i^n the set of all length i subsequences of $[1..n]$ and by A_{IJ}, $I \in S_i^n$, $J \in S_i^n$, the $i \times i$ determinant of the submatrix of A in the rows I and columns J.*

Using the Cauchy-Binet formula [Gantmacher, 1959, Proposition I.§2.14], note that v_c, as a coefficient of the characteristic polynomial, satisfies for some i:

$$\pm v_c = \sum_{I \in S_i^n} B_{II} = \sum_{I \in S_i^n} \sum_{K \in S_i^n} A_{IK} A_{KI}^t = \sum_{I \in S_i^n} \sum_{K \in S_i^n} A_{IK}^2.$$

As a sum of squares, such a coefficient is nonzero if and only if some A_{IK} is nonzero. Hence v_c is the coefficient for the case $i = \text{rank}(A)$. Moreover, if p occurs in the Smith form, then $p|gcd(A_{IK})$, the r-th determinantal divisor of A. It follows that $p^2|v_c$. □

It is straightforward to show that this theorem holds when $\mathbb{Z}$ is replaced by any principal ideal domain of characteristic zero (in general replacing transpose by conjugate transpose).

Corollary 2.1: *Algorithm* Valence-Smith-Form *correctly computes the Smith Form.*

Proof: The theorem shows that we consider the relevant primes. It is evident that the integer Smith form may be composed from the relevant local Smith forms, since the integer Smith form *is* the local Smith form at p up to multiples which are units mod p. □

The remaining sections are devoted to details, variants, and experiments concerning the valence algorithm (sections §3 and §4 are devoted to details on part 1 of the Valence algorithm and section §5 will focus on part 4).

3. Computing the Valence

The first two steps of the valence algorithm have the purpose of determining a small, at any rate finite, set of primes which includes all primes occurring in the Smith form.

3.1. A Point about the Symmetrization

The choice between A^tA and AA^t is easily made in view of the following.

THEOREM 3.1: minpoly_{A^tA} *and* minpoly_{AA^t} *are equal or differ by a factor of* x.

Proof: Let $m_{A^tA}(X)$ and $m_{AA^t}(X)$ be respectively the minimal polynomials of A^tA and AA^t. The Cayley-Hamilton theorem [Gantmacher, 1959, Theorem IV.§4.2] states that $m_{A^tA}(A^tA) = 0$. Then, by multiplying on both sides by A and A^t we have $Am_{A^tA}(A^tA)A^t = 0$ which means $(Xm_{A^tA})(AA^t) = 0$. Since m_{AA^t} is the minimal polynomial of AA^t it follows that $m_{AA^t}|Xm_{A^tA}$. We can similarly prove that $m_{A^tA}|Xm_{AA^t}$. Then either $m_{AA^t} = Xm_{A^tA}$ or $m_{AA^t} = m_{A^tA}$ or $Xm_{AA^t} = m_{A^tA}$. □

Thus the difference of degree has a negligible effect on the run time of the algorithm. It is advantageous to choose the smaller of AA^t and A^tA in the algorithm, to reduce the cost of the inner products involved. Moreover any bound on the coefficients of minpoly_{A^tA} can then be applied to those of minpoly_{AA^t} and vice versa.

3.2. Chinese remaindering

We compute the integer minimal valence, v, of a matrix B (the valence of its minimal polynomial over the integers) by Chinese remaindering valences of its minimal polynomials mod p for various primes p. The algorithm has three steps. First compute the degree of the minimal polynomial by doing a few trials modulo some primes. Then compute a sharp bound on the size of the valence using this degree. End by Chinese remaindering the valences modulo several primes.

The first question is how many primes must be used for the Chinese remaindering. Using Hadamard's inequality [Gathen and Gerhard, 1999, Theorem 16.6] would induce a use of $O(n)$ primes. We found several methods to reduce this number. In the two next sections we develop two methods for this purpose. First is an early termination of the Chinese remaindering, which is directly useful for sequential computation. Then, for a deterministic computation, it is interesting to have a sharper estimate. We can use the ovals of Cassini to bound the spectral radius and thence the valence. We end this section by considerations on probabilistic computations of the integer minimal polynomial.

3.3. Early termination

In the computations for which timings and results are reported here, we compute v_{p_i}, the minimal valence mod p_i, for several primes chosen at random in the vicinity of 2^{16}. We accept the result of Chinese remaindering when the product of the primes p_i exceeds the smaller of the Hadamard bound or a bound computed by considering ovals of Cassini as discussed in section 3.4. But when both of these bounds are large, we use a probabilistic termination condition. Let v be the valence and let v_k be v reduced mod $M = \prod_{i=1}^{k} p_i$ for randomly chosen primes p_i. Thus $0 \leq v_k < M$ and $v_k = v \bmod M$,. Suppose at this point that we believe $v_k = v$, that is to say we believe to have sufficiently many primes even if M is lower than our bound (we may believe this for instance if this v_k remains the same for successive k). Then it is possible to compute a quick check of this belief in the following manner. Choose another random prime p^* in a sufficiently large set and compute v^*, the minimal valence mod p^*. Thus $0 \leq v^* < p^*$ and $v^* = v \bmod p^*$. Also reduce $v_k \mod p^*$, $0 \leq v_k^* < p^*$ and $v_k^* = v_k \bmod p^*$. then if $v_k = v$, the two values modulo p^* must also be equal. Now there are two cases. On the one hand if those two values are distinct we know that our computation is not finished. On the other hand if the values modulo p^* are equal then v_k is v with a high probability which we make explicit in the following lemma.

Lemma 3.1: *Let $v \in \mathbb{Z}$ and an upper bound U be given such that $v < U$. Let P be a set of primes and let $\{p_1 \dots p_k, p^*\}$ be a random subset of P. Let l be a lower bound such that $p^* > l$ and let $M = \prod_{i=1}^{k} p_i$. Let $v_k = v \mod M$, $v^* = v \mod p^*$ and $v_k^* = v_k \mod p^*$ as above. Suppose now that $v_k^* = v^*$. Then $v = v_k$ with probability at least $1 - \frac{\log_l(\frac{U-v_k}{M})}{|P|}$.*

Proof: The only way to give an incorrect answer is to have $v \neq v_k$ and at the same time $v_k^* = v^*$. This means that $v_k = v \mod M$ and $v_k = v \mod p^*$ and therefore, as M and p^* are coprime, p^* must divide $\frac{v-v_k}{M}$. To finish we see that there are at most $\log_l(\frac{v-v_k}{M})$ distinct prime numbers greater than l dividing this quotient and that $v - v_k < U - v_k$. □

For instance, the worst example given in table 2 is a matrix for which the valence is bounded by $U = 117^{827}$. We choose some primes greater than $l = 2^{15}$. We can suppose that M, the product of primes, is greater than 2, therefore $\log_l(\frac{U}{M}) \leq 379$. On the other hand, we know that there are exactly 3030 primes between 2^{15} and 2^{16}. Therefore by choosing a prime between 2^{15} and 2^{16} we still have more than 87% chance of being right and by using this trick four times this grows to 99.97%. Usually, for the homology matrices the bound is closer to 10^{200}. There, only one application of the trick gives more than 98.5% confidence.

3.4. Ovals of Cassini

For a parallel computation of the valence, in particular, or to improve the probability of success, a sharp bound on the valence is very useful. The Hadamard bound may be used, but is too pessimistic an estimate for many sparse matrices. Therefore, we use a bound determined by consideration of Gershgörin disks and ovals of Cassini. This bound is of the form β^d where β is a bound on the eigenvalues and d is the degree of the minimal polynomial.

The i-th Gershgörin disk is centered at $a_{i,i}$ and has for a radius the sum of the absolute values of the other entries on the i-th row. Gershgörin's theorem is that all of the eigenvalues are contained in the union of the Gershgörin disks [Brauer, 1946; Taussky, 1948; Golub and Van Loan, 1996]. One can then go further and consider the ovals of Cassini [Brauer, 1947; Brualdi and Mellendorf, 1994; Varga, 2000], which may produce sharper bounds. For our purposes here it suffices to note that each Cassini oval is a subset of two Gershgörin circles, and that all of the eigenvalues are contained in the union of the ovals. We can then use the following proposition to bound the coefficients of the minimal polynomial:

PROPOSITION 3.1: *Let* $B \in \mathbb{C}^{n\times n}$ *with its spectral radius bounded by* β. *Let* $\text{minpoly}_B(X) = \sum_{k=0}^{d} m_i X^i$. *Then* $|\text{valence}(B)| \leq \beta^d$, *and* $\forall i \in [0..d]$, $|m_i| \leq \max\{\sqrt{d\beta};\beta\}^d$.

Proof: It suffices to note that the valence is a product of d eigenvalues and that $|m_i| \leq \binom{d}{i}\beta^i$ [Mignotte, 1989, Theorem IV.§4.1], and then bound each one of these with either β^d when $\beta \geq d$ or $d^{\frac{d}{2}}\beta^{\frac{d}{2}}$ when $\beta \leq d$. □

For matrices of constant size entries, both β and d are $\mathcal{O}(n)$. However, when d and/or β is small relative to n (especially d) this may be a striking improvement over the Hadamard bound since the length of latter would be of order $n\log(n)$ rather than $d\log(\beta)$.

This is the case for the Homology matrices in our experiments. Indeed, for those, AA^t has very small minimal polynomial degree and has some other useful properties which limit β (e.g. the matrix AA^t is diagonally dominant).

There remains to compute the bound on the spectral radius. We remark that it is expensive to compute any of the bounds mentioned above while staying strictly in the black box model. It seems to require two matrix vector products (with A) to extract each row or column of B. But, if one has access to the elements of A, a bound for the spectral radius of B can easily be obtained with very few arbitrary precision operations:

```
Algorithm: OCB [Ovals-of-Cassini-Bound]
Input:  – a matrix $A \in \mathbb{C}^{m\times n}$.
Output: – $\beta \in \mathbb{R}$, such that for every eigenvalue $\lambda$ of $AA^t$, $|\lambda| \leq \beta$.

(1)    [Centers]
       $\forall i \in [1..m]$, set $q_i = \sum_{j\in[1..n]} a_{ij}^2$.

(2)    [Radii]
       Form $|A|$, the matrix whose entries are the absolute values of those of $A$.
       Compute $v = |A||A|^t[1,1,\ldots,1]^t$.
       $\forall i \in [1..m]$, set $r_i = v_i - |q_i|$.

(3)    [Gershgörin bound]
       Set $q = \max_{i\in[1..m]} |q_i|$.
```

```
        Set $i_1$ such that $r_{i_1} = \max_{i\in[1..m]} r_i$.
        Set $i_2$ such that $r_{i_2} = \max_{i\in[1..m]\setminus\{i_1\}} r_i$.

(3)     [Return Cassini bound]
        Return $\beta = q + \sqrt{r_{i_1} r_{i_2}}$.
```

For a matrix $A \in \mathbb{C}^{m\times n}$ let $\Omega = \max\{m; n;$ number of nonzero elements in $A\}$. Then 2Ω bounds the number of field operations for the matrix vector product, Ax, and for a vector inner product, $x^T x$.

Theorem 3.2: *Let $A \in \mathbb{C}^{m\times n}$ with Ω as described above. Algorithm* Ovals-of-Cassini-Bound *correctly computes a bound on the eigenvalues of AA^t, using no more than 7Ω field operations and $3m$ comparisons.*

Proof: For the correctness of the bound we use the fact that the eigenvalues lie in the union of the ovals of Cassini. Now suppose that q_1, q_2, r_1, r_2 are the two centers and two radii of such an oval. Then any point λ of this oval satisfies the following: $|\lambda - q_1||\lambda - q_2| \leq r_1 r_2$ [Brauer, 1947, Theorem 1]. We want to know the maximal absolute value of such a λ. First, if $|\lambda - q_2| \leq |\lambda - q_1|$, then $|\lambda - q_2| \leq \sqrt{r_1 r_2}$, as $|\lambda| - |q_2| \leq |\lambda - q_2|$, we conclude by $|\lambda| \leq |q_2| + \sqrt{r_1 r_2}$. Replacing q_2 by q_1, the second case is analogous. Therefore β as in the algorithm matches the requirements. The complexity analysis is straightforward. □

3.5. Bad primes and degree of the minimal polynomial

We begin with some definitions:

Definitions 3.1: *Let B be a matrix in $F^{n\times n}$*

- *The* **Krylov subspace** *related to B and a vector $u \in F^n$ is the vector subspace generated by the products of powers of B by u : $Krylov(u, B) = K(u, B) = \text{span}\{u, Bu, B^2u, \ldots\} = \text{span}\{u, Bu, B^2u, \ldots, B^{n-1}u\}$. By extension, the Krylov subspace of a non-square matrix is the Krylov subspace related to the square matrix obtained by addition of zero-columns or zero-rows.*
- *The* **minimal polynomial** *of a vector u related to B is* $\text{minpoly}_{u,B}(x)$, *the monic polynomial of minimal degree annihilating u. By extension, the* **minimal polynomial** *of a subspace S related to B,* $\text{minpoly}_S(x)$, *is the monic polynomial of minimal degree annihilating all the vectors of S.*

The domain of entries for these definitions may be $\mathbb{Z}$ or $GF(q)$. When the domain is not clear from context we add q or $\mathbb{Z}$ to the parameter list, eg. $\text{minpoly}_{u,B,q}$.

It is well known that the degree of the minimal polynomial of a vector is the dimension of its associated Krylov subspace and that the minimal polynomial of a subspace is the least common multiple of the minimal polynomials of the vectors in a basis.

To compute the minimal polynomial of a matrix modulo primes we use Wiedemann's probabilistic algorithm. In order to complete the valence computation we must be sure of the degree of this polynomial over the integers. To compute this degree, we choose some primes at random. The degree of the integer minimal polynomial will be the maximal degree of the minimal polynomials mod p with high probability. Some primes may give a lower degree minimal polynomial. We call them *bad primes*. We next bound the probability of choosing a bad prime at random, by bounding the size of a minor of the matrix that such a prime must divide.

Let δ be the degree of the integer minimal polynomial. There exists a vector u such that the Krylov subspace, $Krylov(u, B)$, associated to B and u is of rank δ. This fact can be easily proved by consideration of the rational canonical form of B. Therefore there exists a square $\delta \times \delta$ nonzero minor, M_δ, of the matrix $[u, Bu, \dots, B^{n-1}u]$. Bad primes must divide this minor. Given an upper bound on M_δ we can then give an upper bound U on the number of bad primes. Let β be an upper bound to the norm of the rows of B. On the one hand, using Hadamard's inequality ([Gathen and Gerhard, 1999, Theorem 16.6]), we can state that

$$|M_\delta| \leq ||u||.||Bu|| \dots ||B^\delta u|| \leq \beta^{\frac{\delta^2}{2}} ||u||.$$

On the other hand, Ozello proved that there exists a vector u with entries less than $\lceil \frac{\delta}{2} \rceil$ such that its minimal polynomial is that of B, $\text{minpoly}_{u,B} = \text{minpoly}_B$ [Ozello, 1987, Theorem III.4.a]. We can therefore bound $||u||$ by $\lceil \frac{\delta}{2} \rceil \sqrt{n}$ and finally state that

$$|M_\delta| \leq \lceil \frac{\delta}{2} \rceil \sqrt{n} \beta^{\frac{\delta^2}{2}} = U.$$

Suppose we choose primes at random from a set P of primes each greater than a lower bound l. There can be no more than $\log_l(U)$ primes greater than l dividing M_δ. It suffices to pick from an adequately large set P to reduce the probability of choosing bad primes. The distribution of primes assures that adequately large P can be constructed containing primes that are not excessively large. For instance, we know these bounds on the k-th prime, p_k :

$$k\big(\ln(k) + \ln(\ln(k)) - 1\big) \leq p_k \qquad k \geq 2 \qquad (1)$$

$$p_k \leq k\left(\ln(k) + \ln(\ln(k)) - 1 + 1.8\frac{\ln(\ln(k))}{\ln(k)}\right) \qquad k \geq 13 \qquad (2)$$

$$p_k \leq k\big(\ln(k) + \ln(\ln(k)) - 0.9427\big) \qquad k \geq 15985 \qquad (3)$$

$$p_k \leq k\left(\ln(k) + \ln(\ln(k)) - 1 + \frac{\ln(\ln(k)) - 1.8}{\ln(k)}\right) \qquad k \geq 27076 \qquad (4)$$

Inequality (1) is from [Dusart, 1999] and (2), (3), (4) are from [Massias and Robin, 1996, Theorem A].

Now, it is of great importance to reduce this U in order to pick small primes for the computations. The bound depends on the size of the vector u and on δ; and since δ is unknown, we only can bound it by n. Therefore $\log_l(U)$ can be quite large, $(O(n^2))$ in practice. We will next show that there exist a vector u with small entries, and that, depending on preliminary computations, we can bound U using a computed degree δ which may be much smaller than n.

In order to prove that fact we need a generalization of Ozello's theorem [Ozello, 1987, Theorem III.4.a] to bound the size of the coefficients of a vector which has a minimal polynomial of *at least* a certain degree:

LEMMA 3.2: *Let B be a symmetric matrix in $\mathbb{Z}^n$ with minimal polynomial of degree δ. For $d \leq \delta$, there exists a vector u with integer entries of absolute value less than $\lceil \frac{d}{2} \rceil$ such that* $\text{minpoly}_{u,B}$ *is of degree at least d.*

Proof: Consider $u = [U_1, U_2, \ldots, U_n]$ as a vector of indeterminates and form the matrix polynomial $C_d(U_1, U_2, \ldots, U_n)$ with columns $u, Bu, \ldots, B^{d-1}u$. We know that there exists a vector $u_0 \in \mathbb{Z}^n$ such that the Krylov subspace, $Krylov(u_0, B)$, associated to B and u_0 is of rank δ. Now as $d \leq \delta$, there also must exist a non identically zero $d \times d$ minor of $C_d(U_1, U_2, \ldots, U_n)$. This minor is a homogeneous polynomial in the U_i, of total degree d. By the Zippel-Schwartz lemma (see [Zippel, 1993] or [Gathen and Gerhard, 1999, Lemma 6.44]), this minor cannot have more than ds^{n-1} zeroes in S^n where $S \subset \mathbb{Z}$ is a set of size s, First if we choose $s = d+1$, we see that there are at most $d(d+1)^{n-1}$ zeroes for this polynomial in a lattice with $(d+1)^n$ elements. We may take S to be the set of integers of absolute values less than $\lceil \frac{d}{2} \rceil$. Then there must exist a vector $u_{[d+1]}$ in S^n for which the minor is nonzero. This vector has a minimum polynomial of degree at least d. Otherwise $C_d(u_{[d+1]})$ could not be of rank d. □

Indeed with $d = \delta$ we have another proof of Ozello's theorem.

3.6. Integer Minimal Polynomial and Valence

We now give the complete algorithm for the computation of the Valence, ending the section with the probabilistic analysis. The algorithm involves computation of minimal polynomials over Z_p. For the fast probabilistic computation of these we use Wiedemann's method (and probability estimates) [Wiedemann, 1986] with early termination as in [Kaltofen et al., 2000]. We then construct the integer minimal polynomial using Chinese remaindering.

In the following we will denote by $\mu_l(x)$ a lower bound on the number of distinct primes between l and x; this bound is easily computed using reciprocals of inequalities (2), (3), (4), and direct bounds on $\pi(x)$, the number of primes lower than x, [Dusart, 1998, Theorem 1.10]:

$$\pi(x) \leq \frac{x}{\ln(x)}\left(1+\frac{1.2762}{\ln(x)}\right) \qquad x \geq 1 \qquad (5)$$

$$\pi(x) \leq \frac{x}{\ln(x)-1.1} \qquad x \geq 60184 \qquad (6)$$

$$\pi(x) \leq \frac{x}{\ln(x)}\left(1+\frac{1}{\ln(x)}+\frac{2.51}{\ln^2(x)}\right) \qquad x \geq 355391 \qquad (7)$$

$$\pi(x) \leq \frac{x}{\ln(x)}\left(1+\frac{1.0992}{\ln(x)}\right) \qquad x \geq 13320000000 \qquad (8)$$

Indeed, we want to have $\mu_l(x) \leq \pi(x) - \pi(l)$. Therefore, we compute an upper bound $\eta(l)$ of $\pi(l)$ with inequalities (5), (6), (7) and (8). Then, in some cases, direct lower bounds for $\pi(x)$ ([Dusart, 1998]) can be used or, in general, an integer k such that $p_k \leq x$ is computed (via Newton's iteration for instance) using inequalities (2), (3) or (4). Now, as $p_k \leq x$, we have $k \leq \pi(x)$ and we conclude with $\mu_l(x) = k - \eta(l)$.

Algorithm: `IMP` [`Integer-Minimal-Polynomial`]

Input:
- a matrix A in $\mathbb{Z}^{n\times n}$.
- an error tolerance ϵ, such that $0 < \epsilon < 1$.
- an upper bound m on primes for which computations are fast, $m > 2^{15}$.

Output:
- the integer minimal polynomial of A, correct with probability at least $1-\epsilon$.

```
(1)   [Initialization, first set of primes]
      set $l = 2^{15}$;
      set $d = 0$; set $F = \emptyset$; set $P = \emptyset$;
      $\beta =$ Ovals of Cassini Bound of $A$;
      set $M = m$;                                  [ Computations will be fast ]

(3)   [ Compute polynomials modulo $p_i$ ]
      Do
        Choose a prime $p_i$, with $l < p_i < M$.        [ at least $\mu_l(M)$ of those ]
        Compute polynomial $w_{A,p_i}$ by Wiedemann's method.
                        [ $w_{A,p_i} =$ minpoly$_{A,p_i}$ with probability at least $1-\frac{1}{p_i}$]
        if $deg(w_{A,p_i}) > d$ then
          set $d = deg(w_{A,p_i})$; set $F = \{p_i\}$; set $P = \{w_{A,p_i}\}$;
          set $U = \sqrt{n}\lceil\frac{d}{2}\rceil\beta^{\frac{(d+1)^2}{2}}$;
          set $bad = \log_l(U)$;                  [ At most that many bad primes ]
```

```
    set $b_i = 2 \times bad(1 + \frac{2}{l-2}) + 3512$ ;          [3512 primes $< 2^{15}$]
    set $M_i$ = upper bound for $p_{b_i}$ ;          [ At least $b_i$ primes are $< M_i$]
    if $(M_i > M)$ then
      set $M = M_i$;                          [ Computations will be slower,
                  degree will be correct with probability at least $\frac{1}{2}$]
    endif
  else, if $deg(\text{minpoly}_{A,p_i}) = d$ then
    $F = F \cup \{p_i\}$;
    $P = P \cup \{\text{minpoly}_{A,p_i}\}$;
  endif
While $\prod_F p_i < \max\{\sqrt{d\beta}; \beta\}^d$ or $\epsilon < \prod_F(\frac{1}{p_i} + \frac{bad}{\mu_l(M)})$;

(4)   [ Chinese remainders ]
      Return $\text{minpoly}_A = \sum_{j=1}^{d} \alpha_j X^j$, where each $\alpha_j \in \mathbb{Z}$ is built from $P$ and $F$.
```

The binary cost of the multiplication of two integers of lenth n, will be denoted by $I_m(n)$: classical multiplication uses $I_m(n) = 2n^2$ bit operations, Karatsuba's method uses $I_m(n) = \mathcal{O}(n^{1.59})$ and Schönhage & Strassen's method uses $I_m(n) = \mathcal{O}(n \log(n) \log\log(n))$ [Gathen and Gerhard, 1999]. For convenience, we will also use "soft-Oh" notation: for any cost functions f and g, we write $f = \mathcal{O}\tilde{}(g)$ if and only if $f = \mathcal{O}(g \log^c(g))$ for some constant $c > 0$.

Theorem 3.3: *Algorithm* Integer-Minimal-Polynomial *is correct.*
Let $s = d \max\{\log_2(\beta(A)); \log_2(d)\}$*, which bounds the lengths of the minimal polynomial coefficients. The algorithm uses expected time*

$$\mathcal{O}(sd\Omega \log(\epsilon^{-1}))$$

for constant size entries. It uses $\mathcal{O}(n \log(s) + ds)$ *memory if every coefficient of the minimal polynomial is computed, and* $\mathcal{O}(n \log(s) + s)$ *if only the valence is computed. The latter is also* $\mathcal{O}\tilde{}(n)$*.*

Proof: For the correctness, the first issue is to be able to choose small primes, i.e. close to word size, in order to use fast computations. But the bound given in section 3.5 might be too large because of its n^2 exponent. However, it is possible to start to compute with small random primes and readjust this bound as some degrees are computed. Indeed, consider again the vector u such that the Krylov subspace associated to B and u, $Krylov(u, B)$ is of rank δ, the degree of the integer minimal polynomial, and suppose we picked a prime p producing a degree d polynomial. Then $Krylov(u, B)$ is of rank $d \mod p$, and therefore p must divide a $(d+1) \times (d+1)$ minor in the first $d+1$ columns. There are at most $\beta^{\frac{(d+1)^2}{2}}$ such primes. We can then sharpen the bound on the size of the primes from $\frac{n^2}{2}\log(\beta)$ to $\frac{(d+1)^2}{2}\log(\beta)$. The next issue is the ending of the loop. The first member of the stopping condition ensures to have enough primes to Chinese remainder the coefficients. When computing only the valence this can be reduced from $\max(\sqrt{d}\beta, \beta)^d$ to β^d. The second member is to have a sufficiently large probability of success: having $|F|$ polynomials of the same degree means that either they are all correct or they are all wrong. Moreover the probability that any one of them is wrong is no more than the probability that Wiedemann's algorithm failed, $\frac{1}{p_i}$, plus the probability that p_i was a bad prime, which is bounded by the number of bad primes over the total number of primes in our set.

Now consider the memory complexity. On the one hand, the valence is bounded by β^d. To store it and the primes, or equivalently the remainders of the valence and their associated primes, we need $\mathcal{O}(s)$ memory; $\mathcal{O}(ds)$ for the whole polynomial. On the other hand, Wiedemann's algorithm uses only a constant number of extra polynomials and vectors over the prime fields. The primes are bounded by M, therefore they are of size $\log_2(M) \leq \log_2(p_{b_i})$. This size, computed with inequalities (2), (3) and (4), is therefore $\mathcal{O}(\log(b_i(\ln(b_i)+\ln\ln(b_i))) = \mathcal{O}(\log(b_i))$. Using b_i as in the algorithm, we conclude that $\mathcal{O}(\log(b_i)) = \mathcal{O}(\log(d^2\log_l(\beta))) = \mathcal{O}(\log(s))$. Each vector being of size n the amount of space needed to store them is $\mathcal{O}(n\log(s))$. At each new Wiedemann's algorithm call, we keep only the remainders of the minimal polynomial and their associated primes. The space allocated for the vectors during the preceding call is reused. Therefore the overall memory cost remains $\mathcal{O}(ds + n\log(s))$ for the whole polynomial and $\mathcal{O}(s + n\log(s))$ for the valence only. The "soft-Oh" complexity is deduced from the latter as $s = d\max\{\log_2(\beta(A)); \log_2(d)\} = \mathcal{O}\tilde{}(d) = \mathcal{O}\tilde{}(n)$.

We complete the proof with the expected time analysis. On the one hand, for each iteration Wiedemann's method requires $\mathcal{O}(d\Omega)$ operations on its ground field; every one of these using at most $I_m(\log_2(p_i)) = \mathcal{O}(\log_2(p_i))$ bit operations. The overall cost of the Wiedemann's iterations is then $\mathcal{O}(\sum d\Omega\log(p_i)) = \mathcal{O}(d\Omega s)$ as s bounds the size of the integer coefficients. On the other hand, as

$$\frac{1}{p_i} + \frac{bad}{\mu_l(M)} \leq \frac{1}{l} + \frac{bad}{2bad(1+\frac{2}{l-2})} = \frac{1}{2},$$

the success probability of $1-\epsilon$ is achieved with at most $\frac{-\log(\epsilon)}{-\log(0.5)} = \mathcal{O}(\log(\frac{1}{\epsilon}))$ good primes. Now for each iteration, Wiedemann's polynomial is correct with probability at least $\frac{1}{2}$. The expected number of iteration will thus not exceed twice the wanted number of good primes. The complexity of this part is then

$$\mathcal{O}(sd\Omega\log(\epsilon^{-1})).$$

Last, the Chinese remaindering cost is negligible. Indeed $\mathcal{O}(I_m(s)log(log(s)))$ operations are needed for each coefficient of the minimal polynomial, cf. [Gathen and Gerhard, 1999, Theorem 10.25]. The latter is $\mathcal{O}\tilde{}(ds)$ for the whole polynomial and $\mathcal{O}\tilde{}(s)$ for the Valence only. □

In practice the actual number of distinct primes greater than 2^{15} dividing valences of homology matrices is very small (no more than 50, say) and we often picked primes between 2^{15} and 2^{16} where there are 3030 primes. This giving us, at most, only 1.7% of bad primes. With only 10 polynomials this reduces to a probability of failure less than 2×10^{-16}.

4. Reducing the prime set : Null Space Method

Consider a prime p which occurs in the Smith Form of A. We know that p^2 divides the characteristic valence of AA^t. It seems more likely in general that p^2 divides the minimal valence than that it divides two or more successive invariant factors of the characteristic polynomial. Of course one can construct examples to the contrary. For instance consider $A = \left[\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right]$, which has Smith form $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 2 \end{smallmatrix}\right]$. However $AA^t = A^tA = \left[\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right]$ and has minimal polynomial $x - 2$ and characteristic polynomial $(x-2)^2$. At any rate, in the boundary matrix examples from homology that we examined we have never encountered a prime occurring singly in the valence which actually occurs in the Smith form of A.

Thus we take as the next goal after the valence computation in algorithm VSF, to determine for a prime p occurring singly in the valence, if the rank mod p is the integer rank. Moreover we would like to be able to decide this as quickly as possible. This job may be done by computing the rank mod p via Wiedemann's algorithm as is discussed below or via elimination. However the run time of those methods is a function of the rank. We therefore propose here a method with run time a function of the degree of the minimal polynomial of $B = AA^t$. However this method requires arbitrary precision integer arithmetic while the rank mod p approach does not. Despite this, when the rank is large relative to this degree, this method is likely to be less costly.

The idea is to use an irreducible factor R of $M = \text{minpoly}(B)$ such that $M = RN$. We would like to know if this factor is repeated in the Frobenius normal form of A, i.e. if the dimension of the kernel of $R(B)$ is the degree of R or is a multiple of this degree. We will show in the following lemma that in the case where R and N are coprime, the dimension of the span of $N(B)$ is equal to the dimension of the kernel of $R(B)$. This leads to a probabilistic algorithm: For $d = \deg(R)$, we try $d+1$ random vectors u_i and see if the $v_i = N(B)u_i$ are dependent. Then if the vectors are dependent we know that with high probability the dimension of the kernel of $R(B)$ is d and that R is not repeated. It follows that any prime occurring singly in valence(R) and valence(M) will occur also singly in the characteristic valence of AA^T. And then such a prime cannot appear in the Smith form of A.

We now give the complete algorithm, then the dimension lemma and end this section with the probabilistic analysis.

Algorithm: `NSD` [`Null-Space-Dimension`]

```
Input:   – a matrix $A \in \mathbb{Z}^{m\times n}$, $A$ may be a Black Box.
         – the minimal polynomial $M$ of $A^tA$, and a factor $R$ of $M$, irreducible
           over $\mathbb{Q}$.
         – $\epsilon \in \mathbb{R}$ such that $0 < \epsilon < 1$.
Output:  – a list $\overline{L}$ of all the primes in valence($R$) which do not occur in the
           Smith form of A. The list is correct with probability at least $1 - \epsilon$.

(1)      [ Initializations ]
         Set $\overline{L} = \emptyset$
         Set $d = \deg(R)$.
         Set $N = \frac{M}{R}$
         Set $g = \lceil \epsilon^{-\frac{1}{d}} \rceil$.
         Set $q$ a random prime such that $q > g$ and $q \nmid$ valence($M$).
         Form the Black Box $N(B)$.

(2)      [ Probabilistic Null-Space dimension ]
         Select $d + 1$ random nonzero vectors $u_i \in (\mathbb{Z}_q)^n$.
         $\forall i \in [0..d]$, set $v_i = N(B)u_i$.
         if $rank([v_0, v_1, \dots, v_d]) < d + 1$ modulo $q$.
            for each prime $p$ dividing valence($R$)
               if $p^2$ does not divide valence($M$), add $p$ to $\overline{L}$.

(3)      [ Return not occurring primes ]
         return $\overline{L}$.
```

Of course this algorithm can be applied to any factor of M to determine primes which can be removed from the candidate list L of primes dividing valence(M). To prove its correctness we need the following lemma.

Lemma 4.1: *Let $B \in \mathbb{Z}^{n\times n}$. Let $N, R \in \mathbb{Z}[X]$ be coprime such that $N(B)R(B) = 0 \in \mathbb{Z}^{n\times n}$. Then* span$(R(B))$ = ker$(N(B))$ *and* span$(N(B))$ = ker$(R(B))$.

Proof: First, since $R(B)N(B) = N(B)R(B) = 0$ the span of one matrix polynomial is included in the kernel of the other one. Now, as R and N are coprime, we use [Gantmacher, 1959, Theorem VII.§2.1] which establishes that ker($R(B)$) and ker($N(B)$) are supplementary. We conclude the proof by use of the dimension theorem: dim(span(X)) + dim(ker(X)) = n, for $X \in \mathbb{Z}^{n\times n}$. □

Theorem 4.1: *Algorithm* Null-Space-Dimension *is correct.*

Proof: Let $B = A^tA$. B is symmetric, so its minimal polynomial P is squarefree. We now suppose that this minimal polynomial is not irreducible. Let N and R be two cofactors of P, R being irreducible. As P is squarefree, N and R are coprime. By the lemma, the span of $N(B)$ is the nullspace of $R(B)$. Thus the nullspace of $R(B)$ has dimension kd, where k is the multiplicity of R in the characteristic polynomial of B. Here we use the fact that R is irreducible to obtain better probabilities (kd instead of only $d+1$ if R was reducible). Hence algorithm NSD may give incorrect result only if it's $d+1$ random vectors are preimages of $d+1$ dependent vectors in a space of dimension kd over a field with more than g scalars. Note that the v_i are uniformly distributed in span$(N(B))$ if the u_i are uniformly distributed in $(\mathbb{Z}_q)^n$.

We now quantify the probability of such a dependence. Let $P(j,n)$ be the probability of a dependency among j random vectors in a space of dimension n $(j \leq n)$ over the field $\mathbb{Z}_q$ with q elements. Then

$$P(j,n) = P(j-1,n) + P(\text{ first } j-1 \text{ independent but } j\text{-th dependent})$$

which gives

$$P(j,n) \leq P(j-1,n) + \frac{q^{j-1}}{q^n} \leq \frac{q^j - 1}{(q-1)q^n} < (q-1)^{j-1-n}.$$

Hence, as $q > g$, if $k \geq 2$ then: $P(d+1,kd) \leq g^{d-kd} \leq \frac{1}{g^d} \leq \epsilon$. $\square$

5. Local Smith form at p

Next consider the question of computing the local Smith form in $\mathbb{Z}^{(p)}$. This is equivalent to a computation of the rank mod p^k for sufficiently many k. Recall that we define the rank mod p^k as the number of nonzero invariant factors mod p^k. We do not mean the McCoy rank, the size of the largest nonzero minor mod p^k. In a number of cases, we have had success with an elimination approach, despite the fill-in problem. We first present this elimination method then the iterative method with lower space requirements.

5.1. Elimination Method

Due to intermediate expression swell, it is not effective to compute directly in $\mathbb{Z}^{(p)}$, the local ring at p, so we perform a computation mod p^e, which determines the ranks mod p^k for $k < e$ and hence the powers of p in the Smith form up to p^{e-1}. Suppose by this means we find that s_r is not zero mod p^e, where r is the previously determined integer rank of A. Then we have determined the Smith form of A locally at p. If, however, the rank mod p^e is less than r, we can repeat the LRE computation with larger exponent e until s_r is nonzero mod p^e.

Algorithm: `LRE` [`Local-Ranks-by-Elimination-mod-`p^e]

Input: – a matrix $A \in \mathbb{Z}^{(m+1)\times(n+1)}$, with elements (a_{ij}) for $i,j \in [0..m] \times [0..n]$.
– a prime p.
– a positive integer e.

Output: – the ranks r_{p^i}, for $1 \leq i \leq e$.

```
(1)   [ Initializations ]
      set $k = e$
      set $r = 0$

(2)   [ $e$ successive Gauss steps ]
      for (exponent $k = 1$ to $e$) do
         while ($\exists (s,t) \in [r..m] \times [r..n], p \not| a_{st}$)
            [ $a_{st}$ is the pivot ]
            Swap rows $r, s$, and columns $r, t$
            for all $(i,j) \in [r+1..m] \times [r+1..n]$ do
              [ elimination, with division, mod $p^{e-k+1}$ ]
                set $a_{i,j} = a_{i,j} - a_{r,j} a_{i,r} / a_{r,r} (\mod p^{e-k+1})$
            set $r = r + 1$.
         set $r_{p^k} = r$.
         [ and invariant factors $s_i = p^{k-1}$ for $r_{p^{k-1}} < i \le r_{p^k}$.]
         for all $(i,j) \in [r..m] \times [r..n]$ do
            set $a_{ij} = a_{ij}/p$

(3)   [ Return local ranks ]
      return $r_{p^i}$, for $i \in [1..e]$
```

Theorem 5.1: *For a positive integer e, algorithm* Local Ranks by Elimination mod p^e *is correct and runs using $\mathcal{O}(r\ m\ n)$ arithmetic operations and $\mathcal{O}(r\ m\ n)$ memory mod p^e where r is the rank mod p^e.*

Proof: It is equivalent to consider the case when the row and column permutations are done in advance so that the pivots are already in the (r,r) position in the while loop. For each k, then, we have an elimination phase determining r_{p^k} followed by a division phase. The elimination may be viewed as multiplication by a unit lower triangular matrix, call it's inverse L_k. The division is multiplication by D_k^{-1}, where $D_k = diag(1,\dots,1,p,\dots,p)$, with r_{p^k} 1's. Then in effect, A has been factored as $P \prod_{k=1}^{e} L_k D_k A' Q$, where P and Q are permutations, A' is the upper triangular final form of A after the elimination, and the D's and L's are as above. We note that $\prod_{k=1}^{e} L_k D_k A' = \prod_{k=1}^{e} L_k D_k U$ in $\mathbb{Z}_{p^e}$, where U is the unimodular matrix obtained by replacing the last $n - r_{p^e}$ rows of A' by those of the identity matrix. It follows that A is equivalent to $B = \prod_{k=1}^{e} L_k D_k$. From this it is easily seen that the Smith form of A in $\mathbb{Z}_{p^e}$ is $S = diag(s_i) = \prod_{k=1}^{e} D_k$, because for both B and S and for each j, the leading principal $j \times j$ minor contains the least power of p, namely $\prod_{i=0}^{j} s_i$. To see this for B, we use Cauchy-Binet expansion ([Gantmacher, 1959, Proposition I.§2.14]) on $L_k D_k$: the leading principal minor of $(L_k D_k)$ is the sum of the product of the same size minors of L_k and D_k. Now, as each L_k is a unit lower triangular matrix, B contains as powers of p only those appearing in the D_k. $\square$

5.2. Iterative Methods

5.2.1. Wiedemann's algorithm and diagonal Scaling

For some matrices the elimination approach just described fails due to excessive memory demand (thrashing). It is desirable to have a memory efficient method for such cases. Two iterative methods are proposed for use here. The first one is “off the shelf”. It is to use Wiedemann's algorithm with the diagonal scaling of [Eberly and Kaltofen, 1997] to compute the rank mod p. This scaling ensures with high probability that the minimal polynomial is a shift of the characteristic polynomial, in fact that it is of the form $m(x) = xf(x)$ where $f(0) \neq 0$ and the characteristic polynomial is of the form $x^k f(x)$ for some k. It follows that the rank is the degree of f. For a given ϵ, to do this with probability of correctness greater than $1 - \epsilon$ requires computation in a field of size $\mathcal{O}(n^2/\epsilon)$ [Eberly and Kaltofen, 1997]. If p if insufficiently large, an extension field may be used. To avoid the large field requirement, we may use the technique as an heuristic, computing the wiedemann polynomial over a smaller field. The resulting polynomial, $w(x)$, is guaranteed to be a factor of the true minimal polynomial, so that it suffices to verify that $w(A) = 0$. This may be probabilistically done by choosing a vector v at random and computing $w(A)v$. The probability that $w(A)v$ is zero while $w(A)$ is nonzero is no more than $1/p$, hence repetition of this process $\log_2(\epsilon^{-1})$ times ensures that the rank has been computed correctly with probability no less than $1 - \epsilon$.

This algorithm has much lower memory requirements than elimination, requiring $\mathcal{O}(\Omega)$ field elements, it has better asymptotic time complexity, $\mathcal{O}(d\Omega \log_2(\epsilon^{-1}))$ field operations, and it is effective in practice for large sparse matrices over large fields. However it doesn't give the complete local Smith form at p. In 5.2.2 we propose a p-adic way to compute the last invariant factor of this local Smith form at p. From this one may infer the complete structure of the local Smith form at p in many cases.

5.2.2. The Last Invariant Factor of the Local Smith Form at p

We have not entirely worked out an extension of Wiedemann's approach suitable for computation of the rank mod a power p^e. The method of Reeds and Sloan Reeds and Sloane [1985] can be adapted to compute the annihilator of our matrix in $\mathbb{Z}_{p^e}$. It may be possible to adapt this to the purpose of computing the rank of the matrix in $\mathbb{Z}_{p^e}$. We do not currently know how to do this in a memory efficient way.

In practice we have encountered matrices whose invariant factors are square free. To verify this it suffices to show that the exponent of p is 1 in the last invariant factor (last nonzero Smith form entry). The following method will do this and a little more.

Let A be a matrix of rank r in $\mathbb{Z}^{m \times n}$ whose local Smith normal form at p is $S_p = \text{diag}\,(p^{k_1}, p^{k_2}, \ldots, p^{k_r}, 0, \ldots, 0)$. The problem is to compute the multiplicity $\kappa = k_r$ of p in the last nonzero invariant factor. Since determining whether κ is zero reduces to comparing the rank modulo p and the rank over $\mathbb{Q}$, we assume that $\kappa \geq 1$.

Our purpose is to derive a black-box algorithm with cost linear in κ rather than in $\Sigma_i k_i$, say.

• Invertible matrix case

We assume for the moment that A is $n \times n$ invertible. For a vector x of reduced integer fractions, we define the order $\mathrm{ord}_p(x)$ of x as the largest exponent of p in the denominators of the entries of x. For a random b the solution x to $Ax = b$ satisfies in general $\mathrm{ord}_p(x) = \kappa$. To reduce the cost of computing κ, let us ensure the same property for the order of the first entry of a well chosen system solution. Let v be a nonzero $n \times 1$ vector with first nonzero entry v_I, $1 \leq I \leq n$, and let u be the first canonical vector. Define the $n \times n$ matrix $E(v)$ by:

$$\begin{cases} E_{i+1,i} = 1 \text{ for } 1 \leq i < I \;\text{ and }\; E_{i,i} = 1 \text{ for } I < i \leq n, \\ E_{i,j} = 0, \text{ otherwise.} \end{cases}$$

Lemma 5.1: *Let b and v be two random integer vectors with entries chosen uniformly in $[0, p-1]$. With probability $(1-1/p)^2$, $E(v)+uv^t$ is invertible ($v \neq 0$) and κ is the order of the first entry of the solution y to $A(E(v) + uv^t)^{-1}y = b$. When $v \neq 0$, the order cannot be strictly greater than κ.*

Proof: If $E(v) + uv^t$ is invertible and x denotes the solution to $Ax = b$ then $y = (E(v) + uv^t)x$ and the first entry of y is $y_1 = \Sigma_i v_i x_i$. Let U and V be unimodular transformations such that $S = UAV$ is in Smith form. Then U and V also define bijections on $\mathbb{Z}_p^n$. Thus:

$$\begin{aligned} \mathcal{P}_b &= \mathcal{P}rob_b(\mathrm{ord}_p(x) = \kappa; Ax = b) \\ &= \mathcal{P}rob_b(\mathrm{ord}_p(x) = \kappa; S(V^{-1}x) = Ub) \\ &= \mathcal{P}rob_b(\mathrm{ord}_p(z) = \kappa; Sz = b) \\ &\geq \mathcal{P}rob_b(1 \leq b_n \leq p-1) = 1 - 1/p. \end{aligned}$$

If $\mathrm{ord}_p(x) = \kappa$ then $p^\kappa x \not\equiv 0 \mod p$. With probability $1 - 1/p$ we have also $p^\kappa \sum_i v_i x_i \not\equiv 0 \mod p$ and the order of y_1 is κ. In this latter case v is nonzero and by construction $E(v)+uv^t$ is invertible. This proves the first assertion of the lemma. For the last assertion it suffices to notice that since the v_i's are integers, the order of $\Sigma_i v_i x_i$ is no higher than the order of x. □

Let us now associate to A a new matrix $\tilde{A}$ whose Smith form is related to that of A and such that one entry of the solution to $\tilde{A}x = b$ can be computed fast:

Lemma 5.2: *Let $q \neq p$ be a prime, for $\tau \geq 1$ integer define $\tilde{A} = p^\tau + qA$. Locally at p, the nonzero invariant factors of A and $\tilde{A}$ having exponents strictly lower than τ are the same.*

Proof: The matrices A and $\tilde{A}$ as matrices in $\mathbb{Z}_{p^\tau}^{n\times n}$ are equivalent. $\square$

Lemma 5.3: *Let b be an integer vector with $||b||_\infty \leq ||A||_\infty = \beta$. Any entry of the solution x to $\tilde{A}x = b$ may be computed in*

$$N = n\left(\log_2 n + 2\log_2(p^\tau + q\beta)\right)/(\log_2 q) + 2 \tag{9}$$

multiplications of $\tilde{A}$ times a vector with entries bounded by q and

$$\mathcal{O}\left(n^2(\log n + \log(p^\tau + q\beta))^2\right) \tag{10}$$

additional binary operations. In addition to the matrix storage, the algorithm requires an $\mathcal{O}(n\,(\log n + \log_2(p^\tau + q\beta)))$ bits of storage.

Proof: We apply the algorithm of Dixon Dixon [1982] based on a q-adic expansion of the solution. The matrix $\tilde{A}$ is invertible in $\mathbb{Q}^{n\times n}$ and in $\mathbb{Z}_p^{n\times n}$: $\tilde{A}^{-1} \equiv p^{-\tau} Id \mod q$. The number N of iterations is given by (9) in the lemma. Each iteration consists in dividing by q a vector of dimension n whose entries have absolute values in $\mathcal{O}(n(p^\tau + q\beta))$ and in multiplying by q an integer of absolute value in $\mathcal{O}(q^N)$. Here we have used the fact that the q-adic expansion of only one entry of the solution is computed. The binary cost of one iteration is thus bounded by $\mathcal{O}((n\log(n(p^\tau + q\beta)) + \log(q^N))M(\log q)/(\log q)$ which is also $\mathcal{O}(NM(\log q))$ and gives (10) once multiplied by N iterations. The rational value of the target entry of the solution vector is constructed from its expansion within the same cost. The extra amount of storage needed is $\mathcal{O}(N\log q)$. $\square$

For any integer τ, from lemma 5.1 and lemma 5.2 we may give a randomized algorithm which compares τ to κ with an arbitrary error tolerance $\epsilon > 0$: By repeated random choices of q, b, v, it is possible to produce an algorithm returning κ, with probability of error as low as required. The pre-conditioning of A by $(E(v) + uv^t)^{-1}$ is required in order to restrict the computation to only one component of the solution vectors.

Algorithm: `LIF [Largest Invariant Factor] order`

```
Input:   – $A \in \mathbb{Z}^{n\times n}$ invertible,
         – a prime $p$, an integer $\tau \geq 1$,
         – an error tolerance $\epsilon$, such that $0 < \epsilon < 1$.
Output:  – with probability at least $1-\epsilon$, returns $\kappa$ if $\kappa \leq \tau$ (maybe wrong) and
           reports that $\kappa$ is strictly greater than $\tau$ otherwise (always correct).

(1)   [ Conditioning ]
      Set max_order = 0.
      Choose a prime $q \neq p$.
      Build $\tilde{A} = p^{\tau+1} + qA$.
```

```
(2)   [ Order ]
      For $t$ from 1 to $\lceil \log(\epsilon)/\log(1-(1-1/p)^2)\rceil$ do
         Choose random $b$ and $v$ in $[0, p-1]^n$.
         If $v \neq 0$ construct $E(v) + uv^t$
            Compute $y_1 \in \mathbb{Q}$ the first entry of $y$ such that $\tilde{A}(E(v)+uv^t)^{-1}y = b$.
            If $\text{ord}_p(y_1) > \tau$ then Return("$\kappa >$" $\tau$).
            max_order := max{max_order, $\text{ord}_p(y_1)$}.
      Return("$\kappa =$" max_order).
```

THEOREM 5.2: *The algorithm LIF works as specified, if $\kappa \leq \tau$ then it returns κ with probability at least $1-\epsilon$ or a lower value with probability less than ϵ. If $\kappa > \tau$ then it discovers the latter inequality with probability at least $1-\epsilon$, this result is always correct, or returns a wrong value lower than τ with probability less than ϵ. The cost of the algorithm is bounded by*

$$\mathcal{O}\left(n\Omega(\log n + 2\log(p^\tau + q\beta))^2 \log(\epsilon^{-1})\right)$$

Proof: If $\kappa \leq \tau$ then by lemma 5.2 all the invariant factors of $\tilde{A} = p^{\tau+1} + qA$ are also of order less than κ. By lemma 5.1, the probability that $v = 0$ or that y_1 gives an order strictly less than κ is $1-(1-1/p)^2$. The computed order cannot be strictly greater than κ. After $\lceil \log(\epsilon)/\log(1-(1-1/p)^2)\rceil$ trials, the probability of having a wrong result is less than ϵ. In the same way, if $\kappa > \tau$ *i.e.* the largest invariant factor of $\tilde{A}$ is at least of order $\tau+1$, the algorithm will certify it with probability at least $1-\epsilon$. The certificate is by lemma 5.1 since the computed order cannot be greater than the actual one. The cost bound is immediately derived from (9) and (10) (multiplying by the inverse of $E(v)+uv^t$ requires linear time only) times the number of trials. $\square$

One may see for instance that with 24 random choices for b and v, it is possible with probability more than $1-10^{-6}$ independently of the dimension of the matrix, to certify that $\kappa \geq 2$ for $p = 3$. By constructing orders of number fields, as done in [Giesbrecht, 1997, §5] for Diophantine equations, the number of trials could be reduced up to some increase in time for the applications of lemma 5.3.

• General case

When A is $m \times n$ singular of rank r, unlike in lemma 5.2, the use of the modified matrix $\tilde{A}$ will always introduce new nonzero invariant factors that prevent us from computing κ using algorithm LIF. An alternative way to proceed is to apply the algorithm to an invertible $r \times r$ matrix A_S constructed from A, whose local Smith form is $S_I = \text{diag}\,(p^{k_1}, p^{k_2}, \ldots, p^{k_r})$. Such a matrix A_S may be obtained by conditioning A. We generalize the construction of [Mulders and Storjohann, 1999, Lemma 11]. Let us denote by U and V two unimodular multipliers such that UAV is in Smith form: $UAV = \left[\begin{smallmatrix} S_I & 0 \\ 0 & 0 \end{smallmatrix}\right]$ and consider two preconditioners $P \in \mathbb{Z}^{r\times m}$ and $Q \in \mathbb{Z}^{n\times r}$. If $T \in \mathbb{Z}^{m\times r}$ is the matrix constructed with the first r

columns of U^{-1} and if $W \in \mathbb{Z}^{r\times n}$ is constructed with the first r rows of V^{-1} then: $A = TS_IW$ thus $PAQ = (PT)S_I(WQ)$. We see that if PT and WQ are invertible modulo p, we can take $A_S = PAQ$. Since U and V are unimodular, T and W have rank r modulo p and the condition depends on the choice of P and Q. We have proven:

Lemma 5.4: *There exists two matrices $W \in \mathbb{Z}^{r\times n}$, $T \in \mathbb{Z}^{m\times r}$ only depending on A giving: If $P \in \mathbb{Z}^{r\times m}$ and $Q \in \mathbb{Z}^{n\times r}$ are such that*

$$p \not| \det(PT) \quad \text{and} \quad p \not| \det(WQ) \tag{11}$$

then $A_S = PAQ$ is invertible and its local Smith form at p is the invertible submatrix $S_I = \text{diag}\ (p^{k_1}, p^{k_2}, \ldots, p^{k_r})$ of the local Smith form of A.

For preconditioners satisfying the lemma condition one may consider Toeplitz matrices as used in [Kaltofen and Saunders, 1991, Theorem 2] or in [Giesbrecht, 1997, §5]. Define $\eta = \lceil \log_2 2r(r+1) \rceil$ and let $\Gamma \in \mathbb{Z}[x]$ be a monic polynomial irreducible modulo p, of degree η and with coefficients bounded in absolute value by p. Let $\theta \equiv x \mod \Gamma$ and $\mathcal{V}_p$ be the set of the polynomials of degree less than η with coefficients 0 or 1 over $\mathbb{Z}[\theta]$.

Proposition 5.1: *Giesbrecht [1997] Let $A \in \mathbb{Z}^{m\times n}$ be of rank r, p a prime. Consider as preconditioners $\mathcal{P}$ and $\mathcal{Q}$, an upper and a lower triangular Toeplitz matrix each with unit diagonal and whose other entries are randomly chosen in $\mathcal{V}_p$. The local Smith form of $A_S = \mathcal{P}A\mathcal{Q}$ at p is S_I with probability greater than $1/2$.*

The algorithm LIF may then be applied to A_S or the random choice of the conditioners may be included in the *for loop* of a modified algorithm. In that case Giesbrecht [1997], if $y_1 = \sum_{0\leq l<\eta} y_1^{(l)}\theta^l \in \mathbb{Q}(\theta)$ is the first entry of the solution to $\tilde{A}_S(E(v) + uv^t)y = b$, one looks at the order of $y_1^{(0)}$. The sizes of the integers (numerators and denominators) involved in the computations are a factor $\mathcal{O}(\eta)$ *i.e.* $\mathcal{O}(\log r)$ greater than for the regular case. The cost bounds of lemma 5.3 will be modified by the same factor.

Instead of Toeplitz matrices one may use sparse matrices as in [Wiedemann, 1986, §III] to avoid extension fields. The *Hamming weight* of a matrix is the number of its nonzero entries.

Proposition 5.2: *[Wiedemann, 1986, Theorem 1′] Let $A \in \mathbb{Z}^{m\times n}$ be of rank r, p a prime. A random procedure exists for generating sparse preconditioners $\mathcal{P}$ and $\mathcal{Q}$ with coefficients in $\{0,1\}$ and of total Hamming weight $\mathcal{O}(r\log r)$. With probability at least $1-\epsilon$ they satisfies that the local Smith form of $A_S = \mathcal{P}A\mathcal{Q}$ at p is S_I.*

In any case, one can include the preconditionning in the preceding process; thus only increasing the computation of the matrix-vector products by a factor $\mathcal{O}(log(r))$.

6. Asymptotic analysis

In this section we collect from the previous sections and summarize the bit complexities, memory requirements, and probabilities of correctness of the various parts and variants of our algorithm. As our motivation comes from matrices with entries in $\{-1,0,1\}$, this analysis is for integer matrices with constant size entries. Recall that for a matrix A, we have $\Omega = \max\{n; m;$ number of nonzero elements in $A\}$, r is the integer rank of A, $d = d_{AA^t}$ is the degree of minpoly_{AA^t}, β bounds the eigenvalues of A, and $s \le d\log_2(\beta)$ is the number of primes dividing the valence of minpoly_{AA^t}. Also, figure 1 gives the organization of the algorithm.

Table 1: Valence Smith form algorithm complexities (small valence)

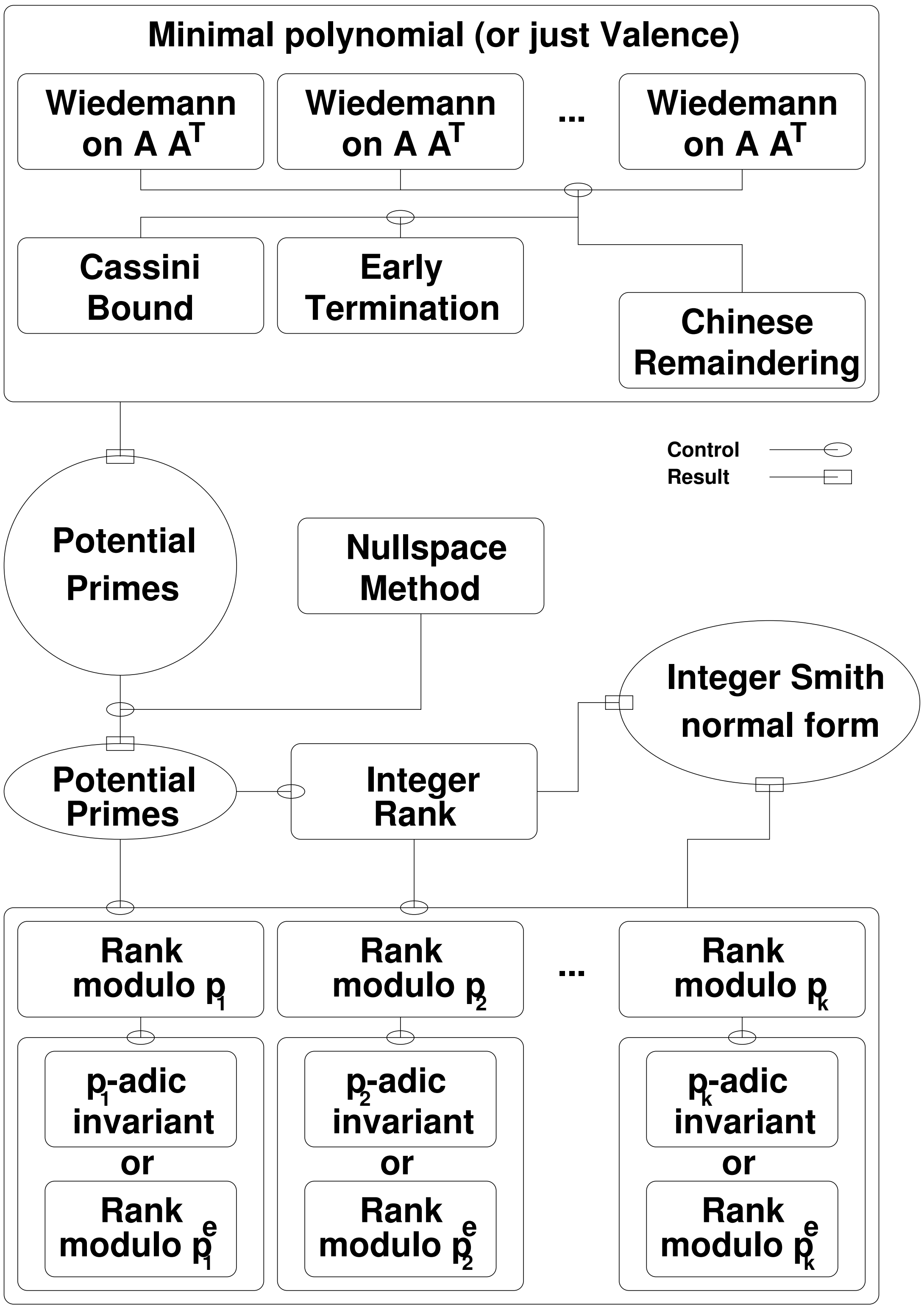
Step	Expected Time		Memory	Probability	Section
	Input dependent	Worst case		of correctness	
Cassini bound	$\mathcal{O}(\Omega)$	$\mathcal{O}(\Omega)$	$\mathcal{O}(n)$	1	3.4
Valence	$\mathcal{O}(sd\Omega\log(\epsilon^{-1}))$	$\mathcal{O}(n^2\Omega\log(\epsilon^{-1}))$	$\mathcal{O}^{\sim}(n)$	$1-\epsilon$	3.6
Elimination techniques					
Integer rank	$\mathcal{O}(rmn)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$1-\epsilon$	2
Prime ranks	$\mathcal{O}^{\sim}(rmn)$	$\mathcal{O}^{\sim}(n^3)$	$\mathcal{O}^{\sim}(n^2)$	1	5.1
Prime Powers	$\mathcal{O}^{\sim}(rmn)$	$\mathcal{O}^{\sim}(n^3)$	$\mathcal{O}^{\sim}(n^2)$	1	5.1
Black Box techniques					
Integer rank	$\mathcal{O}^{\sim}(r\Omega\log(\epsilon^{-1}))$	$\mathcal{O}^{\sim}(n\Omega\log(\epsilon^{-1}))$	$\mathcal{O}^{\sim}(n)$	$1-\epsilon$	5.2.1
Prime ranks	$\mathcal{O}^{\sim}(r\Omega\log(\epsilon^{-1}))$	$\mathcal{O}^{\sim}(n\Omega\log(\epsilon^{-1}))$	$\mathcal{O}^{\sim}(n)$	$1-\epsilon$	5.2.1
Last invariant	$\mathcal{O}^{\sim}(r\Omega\log(\epsilon^{-1}))$	$\mathcal{O}^{\sim}(n\Omega\log(\epsilon^{-1}))$	$\mathcal{O}^{\sim}(n)$	$1-\epsilon$	5.2.2
Overall with					
Elimination	$\mathcal{O}^{\sim}(srmn + d\Omega\log(\epsilon^{-1}))$		$\mathcal{O}^{\sim}(n^2)$	$1-\epsilon$	
Black Box	$\mathcal{O}^{\sim}(sn\Omega\log(\epsilon^{-1}))$		$\mathcal{O}^{\sim}(n)$	$1-\epsilon$	

In this table we assume the valence is small, as it is for the matrices for which this algorithm is particularly effective. We neglect two consequences of a large valence.

First, there can be large prime factors of the valence not rejected by the method of section 4. Existence of large primes could introduce a factor for the cost of arithmetic modulo primes of size $\mathcal{O}(s)$. To include this effect multiply the table entries for prime ranks (elimination and Black Box), prime powers, last invariant, and thus the entries for the overall complexities by the cost of arithmetic modulo a size s prime.

Second we do not show complexities for the integer factorization of the valence. In fact the valence factorization *need not* be precomputed. Indeed it is possible to start the rank computations with the composite valence (using an arithmetic which is $\mathcal{O}^{\sim}(s)$ per operation mod the valence, say). Then in both cases, elimination and black box, problems arising from the nonprimality of the valence will show some of its factors. In the first case, elimination, problems arise when a non-invertible nonzero pivot is found. Then the gcd of this pivot

Figure 1: Valence Algorithm



and the valence reveals two nontrivial factors. Moreover, the elimination can resume modulo both of these factors. Similarly, in the second case, Wiedemann's method, problems arise when a non-invertible nonzero discrepancy is computed. Then again the gcd of this discrepancy and the valence reveals two nontrivial factors and computations can resume modulo both of these factors.

However, for homology matrices, the valence was usually small with small factors and therefore easy to factor using elliptic curves, for instance. We factor as much as we can, which means completely for most of the cases, to isolate small primes, as computations are faster modulo word-sized primes. It is conjectured that the elliptic curve factorization algorithm determines a non trivial divisor of a composite number t in expected time

$$\ln(t)^2 e^{\sqrt{\ln(p)\ln\ln(p)(2+o(1))}}$$

where p is the least prime dividing t [Lenstra, 1987, Conjecture 2.10]. As we will see in table 2, for the cases we considered, s is usually very small with very small least prime divisors; therefore enabling practical performances of the algorithm despite the super-polynomial complexity of the factorization.

7. Experiments with Homology Matrices

In this section we describe the structure of the boundary maps of a simplicial complex. More details on the connection between homology groups of simplicial complexes and linear algebra can be found in [Munkres, 1994]. We will talk mainly about three homology matrix classes. The matrices will be denoted by three naming patterns:

- **mk**i**.b**j denotes the boundary matrix j from the matching complex with i vertices.
- **ch**i**-**k**.b**j denotes the boundary matrix j from the i by k chessboard complex.
- **n**i**c**k**.b**j denotes the boundary matrix j from the not i-connected graph with k vertices.

For details on those simplicial complexes see [Babson et al., 1999; Bjöerner et al., 1994].

The boundary matrices are sparse matrices with a fixed number k of nonzero elements per row and l per column. If A_i is the boundary map between the dimensions i and $i-1$ of a simplicial complex then $k = i+1$. All entries are -1, 0 or 1. Moreover, the Laplacians $A_iA_i^t$ and $A_i^tA_i$ also have -1, 0 and 1 as entries except for the diagonal which is respectively all k and all l. However, as expected, those Laplacians have more than twice as many nonzero elements as A_i. Thus, we did not perform the matrix multiplications to compute $A_iA_i^tv$. We performed two matrix-vector products, $A_i(A_i^tv)$, instead.

We will also check in table 2 that the Laplacians have indeed a very low degree minimal polynomial (say up to 25 for the matching and chessboard matrices, close to 200 for the not-connected). This fact was our chief motivation to develop the Valence method. Figure 2 illustrates the patterns occurring in these matrices. It shows ch4-4.b2, the boundary map between the second and first dimensions of the 4×4 chessboard complex, together with its Laplacians. It is of size 96×72 with 288 nonzero elements. It has 3 elements per row and 4 per column. On the other hand AA^t is of size 96×96 with 960 nonzero elements and A^tA is of size 72×72 with 648 nonzero elements.

Our experiments were realized on a cluster of 20 Sun Microsystem Ultra Enterprise 450 each with four 250 MHz Ultra-II processor and 1024 Mb or 512 Mb of memory. We computed ranks of matrices over finite fields $GF(q)$ where q has half word size. The chosen arithmetic used discrete logarithms with precomputed tables, as in [Sibert et al., 1990]. The algorithms were implemented in C++ with the LINBOX[*] library for computer algebra and the ATHAPASCAN[†] environment for parallelism.

We will call ω the number of nonzero elements per row, $N \times n$ the shape and r the integer rank of the matrix under consideration. We will produce these elements only in table 2. The name of the matrix will be repeated in the following tables. For several cases, fill-in causes a failure of elimination. This is due to memory thrashing (MT). All the timings presented are in seconds except as otherwise specified and reflect the cpu time for sequential computations and the real time for parallel computations.

In table 2 we present computations of the integer minimal polynomial of some homology matrices. We indicate the Cassini bound, CB, for those and present the number of Chinese remainders needed, Rem, the degree of the minimal polynomial, d_m, the computed upper bound for the number of *bad* primes, the associated upper bound for the value of the random primes, M, and the minimal valence, v_m. As some v_m are quite large, we write explicitly only the smaller prime factors and denote by C_i a product of i larger prime factors. In one case, C_{59+} denotes a product of at least 59 primes. This number has 57 known prime factors and a another composite factor of 376 digits that we were unable to factor. We recall that there are respectively $3030, 8746, 19510, 39915$ primes between 2^{15} and 2^{16}, 2^{15} and 2^{17}, 2^{15} and 2^{18}, 2^{15} and 2^{19}, and 560821 primes between 2^{15} and 2^{23}, these results implying the given values for M. We give timings for sequential and parallel computation of the minimal valence. A first approach for parallel computation is to use sequential routines for Cassini bounds, minimal polynomial over $\mathbb{Z}_p$, and Chinese remaindering of integers. The algorithm has 3 steps. First compute the Cassini bound and some minimal polynomials in parallel. Using the bound and the degree of the minimal polynomials, the maximum number of remainders needed is known. Therefore, the second step is the computation in parallel of some more minimal polynomials as required [Dumas, 2000]. The

[*]Symbolic linear algebra library, http://www.cis.udel.edu/~caviness/linbox

[†]Parallel execution support of the APACHE Project, http://www-id.imag.fr/software

Table 2: Computing the Integer Minimal Polynomial of AA^t via Chinese remaindering

Matrix	ω, $N \times n$, r	CB	Rem	d_m	*bad*	M	v_m	Sequential	Parallel
mk9.b3	4, 945x1260, 875	12	2	X^6	11	2^{16}	$2^6.3^4$	0.31	0.39
mk10.b3	4, 4725x3150, 2564,	24	3	X^7	15	2^{16}	$2.3^4.5^3.7.13$	1.16	0.81
mk11.b4	5, 10395x17325, 10143	15	2	X^8	11	2^{16}	$-2^4.3^2.5^2.7.11$	4.00	1.80
mk12.b3	4, 51975x13860, 12440	60	3	X^7	19	2^{16}	$2^{10}.3^5.5.7.11.13$	16.68	5.55
mk12.b4	5, 62370x51975, 39535	30	4	X^{11}	31	2^{16}	$-2^{13}.3^7.5.7.11$	34.49	11.39
mk13.b4	5, 270270x135135, 111463	50	4	X^{11}	35	2^{16}	$-23.2^{11}.3^7.5^2.7.11.13$	149.06	42.13
mk13.b5	6, 135135x270270, 134211	18	4	X^{12}	31	2^{16}	$2^{10}.3^5.5^2.11.13$	96.11	29.41
ch4-4.b2	3, 96x72, 57	12	1	X^4	6	2^{16}	$2^7.3$	0.08	0.24
ch5-5.b3	4, 600x600, 424	16	3	X^8	16	2^{16}	$2^5.3^3.5^2.7$	0.35	0.39
ch6-6.b4	5, 4320x5400, 3390	20	3	X^{10}	23	2^{16}	$2^{10}.3^5.5.11$	1.83	1.43
ch7-7.b4	5, 52920x29400, 22884	45	6	X^{15}	54	2^{16}	$-19.17.2^{11}.3^7.5^3.7^3.11.13$	45.17	10.28
ch7-7.b5	6, 35280x52920, 29448	24	4	X^{12}	33	2^{16}	$-29.41.367.34504396919539$	23.88	7.52
ch7-8.b5	6, 141120x141120, 92959	36	7	X^{21}	92	2^{16}	$19.17.2^18.3^9.5^4.7^3.11.13$	123.18	24.85
ch7-9.b3	4, 105840x17640, 16190	96	7	X^{16}	70	2^{16}	$29.19.2^{19}.3^{10}.5^5.7^4.11.13.31$	82.57	15.04
ch7-9.b4	5, 317520x105840, 89650	75	9	X^{22}	118	2^{16}	$-1433.C_4$	471.22	64.67
ch7-9.b5	6, 423360x317520, 227870	48	9	X^{24}	125	2^{16}	$23.19.17.2^{24}.3^{10}.5^4.7^4.11^2.13^2$	775.33	106.15
ch8-8.b4	5, 376320x117600, 100289	80	7	X^{16}	69	2^{16}	$29.19.17.2^{22}.3^8.5^4.7^2.11.13$	386.13	64.78
ch8-8.b5	6, 564480x376320, 276031	54	7	X^{19}	85	2^{16}	$-19.17.2^{21}.3^9.5^4.7^2.11^2.13$	732.44	127.57
ch8-8.b6	7, 322560x564480, 279237	28	5	X^{14}	44	2^{16}	$2^{15}.3^4.5^3.7.11.13$	337.72	77.52
ch8-8.b7	8, 40320x322560, 40320	8	1	X^1	6	2^{16}	-2^3	10.63	10.47
ch9-9.b3	4, 381024x42336, 39824	144	5	X^{10}	36	2^{16}	$-73.6287.C_2$	190.74	45.10
ch9-9.b4	5, 1905120x381024, ??	125	8	X^{17}	84	2^{16}	$-2^{13}.3^{12}.5^6.7^3.13^2.C_5$	2279.32	358.97
n2c6.b6	7, 5715x4945, 2943	63	25	X^{66}	902	2^{16}	$17.2^{26}.3^{21}.5^2.7.13^2.C_{10}$	55.43	3.44
n2c6.b7	8, 3990x5715, 2772	64	58	X^{152}	4692	2^{18}	$17^2.2^{83}.3^{55}.5^{50}.7^3.13^2.C_{11}$	206.18	7.51
n3c6.b9	10, 2511x4935, 1896	60	15	X^{40}	338	2^{16}	$17^2.2^9.3^{13}.5^8.13^4.7.C_6$	15.80	1.83
n4c5.b5	6, 4340x2852, 1866	60	39	X^{103}	2138	2^{17}	$-2^{16}.3^{10}.5^{16}.13^2.191^2.89^2.C_{21}$	82.76	4.04
n4c5.b6	7, 4735x4340, 2474	63	82	X^{216}	9393	2^{18}	$19.17.2^{45}.3^{10}.5^{26}.7.67^2.59^2.C_{34}$	401.05	11.49
n4c5.b7	8, 3635x4735, 2261	64	100	X^{263}	13952	2^{19}	$-17.2^{47}.3^{13}.5^{28}.11.13^2.59^2.C_{36}$	511.69	14.26
n4c5.b8	9, 1895x3635, 1374	63	62	X^{164}	5434	2^{18}	$17^2.2^{35}.3^{25}.5^{23}.11.13^2.379^2.C_{21}$	125.93	5.30
n4c6.b5	6, 51813x20058, 15228	96	44	X^{104}	2430	2^{17}	$-23.2.52009.C_3$	1161.39	35.35
n4c6.b12	13, 25605x69235, 20165	117	358	X^{827}	157060	2^{23}	$-2^{243}.3^{43}.5^{36}.7^7.11^4.13^2.31^2.C_{59+}$	66344.40	1181.61
n4c6.b13	14, 6300x25605, 5440	98	37	X^{87}	1716	2^{17}	$-2^{31}.3^8.5^8.7^2.11^2.13.31^2.C_{18}$	249.28	9.90

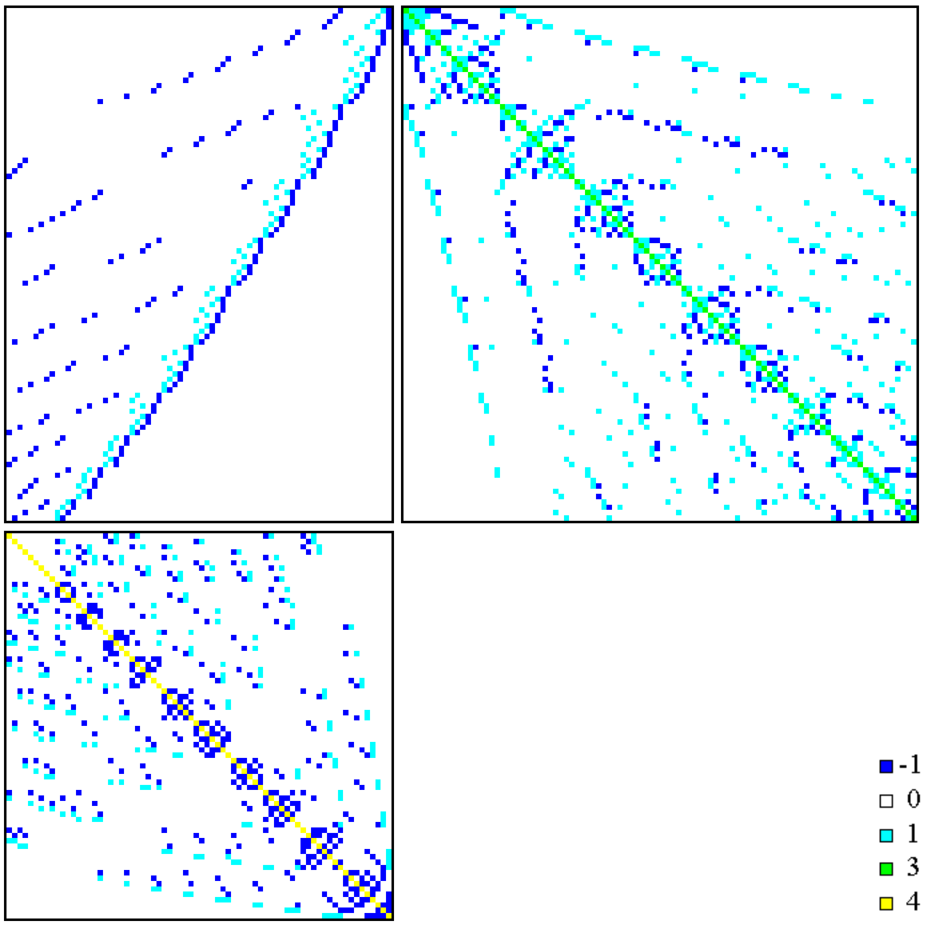


Figure 2: Chessboard complex 4-4, boundary matrix 2 with AA^t and A^tA.

last step is a Chinese remaindering of the coefficients. A future implementation will also use parallel matrix-vector products as well as block methods [Kaltofen, 1995; Villard, 1997] to improve speed.

In table 3 we report some comparisons between Wiedemann's algorithm and elimination with reordering for computing the rank. We just want to emphasize the fact that for these matrices from homology, as long as enough memory is available, elimination is more efficient. However, for larger matrices, Wiedemann's algorithm is competitive and is sometimes the only solution.

In table 4 we compare timings of our algorithm to some implementations of other methods. We compare here only the results obtained using the version of the Valence Smith Form algorithm in which we use Wiedemann's algorithm to compute the Valence and then elimination modulo small powers of primes p to compute the invariant factors locally at p. Simplicial Homology [Dumas et al., 2000] is a proposed GAP share package. It computes homology groups of simplicial complexes via the Smith form of their boundary maps. It features a version of our Valence algorithm as well as an elimination method for homology

Table 3: rank modulo 65521, Elimination vs. Wiedemann

Matrix	Elimination	Wiedemann
mk9.b3	0.26	2.11
mk13.b5	MT	23 hours
ch7-7.b6	4.67	119.53
ch7-6.b4	49.32	412.42
ch7-7.b5	2179.62	4141.32
ch8-8.b4	19 hours	33 hours
ch8-8.b5	MT	55 hours
n2c6.b6	6.44	64.66
n2c6.b7	3.64	49.46
n4c5.b6	2.73	47.17
n4c6.b12	231.34	4131.06
n4c6.b13	8.92	288.57

groups by Frank Heckenbach. The latter is a variant of the classical elimination method over arbitrary precision integers for Smith form [Munkres, 1994], taking advantage of the particular structures of the boundary maps. The entry "Hom-Elim-GMP" in this table refers to this elimination-based method using Gnu Multi Precision integers. Fermat [Lewis, 1997] is computer algebra system for Macs and Windows. Its Smith form routine is an implementation of [Bachem and Kannan, 1979].

Table 4: Fermat vs. Hom-Elim-GMP vs. SFV

Matrix	Fermat	Hom-Elim-GMP	Valence (Eliminations)	
ch6-6.b4	49.4	2.151	27.417	(6)
mk9.b3	2.03	0.211	0.946	(4)
mk10.b3	8.4	0.936	13.971	(7)
mk11.b4	98937.27	2789.707	384.51	(7)
mk12.b3	189.9	26.111	304.22	(7)
mk12.b4	MT	MT	13173.49	(7)

"Hom-Elim-GMP" and "Valence" ran on a 400 MHz sparc SUNW, Ultra-4 processor with 512 Mb, but Fermat is only available on Mac and Windows. We therefore report on experiments with Fermat on a 400 MHz Intel i860 processor with only 512 Mb. First we see that "Fermat" cannot compete with "Hom-Elim-GMP" in any case. The main explanation is that the pivot strategy used by "Hom-Elim-GMP" is very well suited to the homology matrices. We can see also that, as long as no coefficient growth is involved, "Hom-Elim-GMP" is often better than "Valence". Indeed, where "Hom-Elim-GMP" performs only one integer elimination, "Valence" performs an elimination for every prime involved (the number of those eliminations is shown between parenthesis in the column Valence (Eliminations) of the table) - of course in parallel this difference will weaken. But as soon as coefficient growth becomes important "Valence" is winning. Moreover, "Valence" using only memory efficient iterative methods can give some partial results where memory exhaustion due to fill-in prevents any eliminations from running to completion. In table 3 we can see some of these effects and we present some of those partial results in table 5: for some matrices we were able to compute ranks modulo some primes, and therefore the occurrence of these primes in the Smith form, but not the actual powers of these primes. These, however, are the only currently known results about these matrices.

Table 5: Valence Smith Form with Black Box techniques

Matrix	Time	Results	
mk13.b5	98 hours	Partial	: 133991 ones & 220 powers of 3
ch7-8.b5	180 hours	Partial	: 92916 ones & 35 powers of 3 & 8 powers of 2 and 3
ch8-8.b4	264 hours	Complete	: 100289 ones

8. Conclusion

The preceding comparison of two elimination implementations and our Valence method provides a convenient basis for summary remarks.

(1) Elimination can be effective on these sparse but patterned simplicial complex boundary matrices. However this is true only if the pivoting strategy is well suited to this situation.

(2) For large enough sparse matrices, fill-in makes elimination more time consuming than the Valence method, and for the largest examples, elimination fails altogether due to excessive memory demand. With the Valence approach, we were able to compute the rank modulo primes for matrices with 500,000 or more rows and columns, while elimination was failing for matrices of sizes larger than about 50,000.

(3) It remains open how to efficiently determine the ranks modulo powers (> 1) of primes while using memory-efficient iterative methods.

References

Eric Babson, Anders Bjöerner, Svante Linusson, John Shareshian, and Volkmar Welker. Complexes of not i-connected graphs. *Topology*, 38(2):271–299, 1999.

Achim Bachem and Ravindran Kannan. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8(4):499–507, November 1979.

Anders Bjöerner, László Lovász, Sinisa T. Vrećica, and Rade T. Živaljević. Chessboard complexes and matching complexes. *Journal of the London Mathematical Society*, 49(1):25–39, 1994.

Anders Björner and Volkmar Welker. Complexes of directed graphs. *SIAM Journal on Discrete Mathematics*, 12(4):413–424, November 1999.

Alfred Brauer. Limits for the characteristic roots of a matrix. I. *Duke Mathematical Journal*, 13:387–395, 1946.

Alfred Brauer. Limits for the characteristic roots of a matrix. II. *Duke Mathematical Journal*, 14:21–26, 1947.

Richard A. Brualdi and Stephen Mellendorf. Regions in the complex plane containing the eigenvalues of a matrix. *American Mathematical Monthly*, 101(10):975–985, December 1994.

John D. Dixon. Exact solution of linear equations using p-adic expansions. *Numerische Mathematik*, 40:137–141, 1982.

Jean-Guillaume Dumas. Calcul parallèle du polynôme minimal entier en Athapascan-1 et Linbox. In *RenPar'2000. Actes des douzièmes rencontres francophones du parallélisme, Besançon, France*, June 2000.

Jean-Guillaume Dumas, Frank Heckenbach, B. David Saunders, and Volkmar Welker. *Simplicial Homology, a (proposed) share package for GAP*, March 2000. Manual (http://www.cis.udel.edu/~dumas/Homology).

Jean-Guillaume Dumas, B. David Saunders, and Gilles Villard. Integer Smith form via the Valence : experience with large sparse matrices from Homology. In [Traverso, 2000], pages 95–105.

Pierre Dusart. *Autour de la fonction qui compte le nombre de nombres premiers.* PhD thesis, Université de Limoges, 1998.

Pierre Dusart. The k^{th} prime is greater than $k(\ln k + \ln\ln k - 1)$ for $k \geq 2$. *Mathematics of Computation*, 68(225):411–415, January 1999.

Wayne Eberly, Mark W. Giesbrecht, and Gilles Villard. Computing the determinant and Smith form of an integer matrix. In *The 41st Annual IEEE Symposium on Foundations of Computer Science, Redondo Beach, CA*, November 2000.

Wayne Eberly and Erich Kaltofen. On randomized Lánczos algorithms. In [Küchlin, 1997], pages 176–183.

Feliks Rudimovich Gantmacher. *The Theory of Matrices.* Chelsea, New York, 1959.

Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, New York, NY, USA, 1999.

Mark W. Giesbrecht. Probabilistic computation of the Smith Normal Form of a sparse integer matrix. *Lecture Notes in Computer Science*, 1122:173–186, 1996.

Mark W. Giesbrecht. Efficient parallel solution of sparse systems of linear diophantine equations. In *Parallel Symbolic Computation (PASCO'97)*, pages 1–10, Maui, Hawaii, July 1997.

Gene H. Golub and Charles F. Van Loan. *Matrix computations.* Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

Costas 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 on Computing*, 18(4):658–669, 1989.

Erich Kaltofen. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Mathematics of Computation*, 64(210):777–806, April 1995.

Erich Kaltofen, Mukkai S. Krishnamoorthy, and B. David Saunders. Parallel algorithms for matrix normal forms. *Linear Algebra and its Applications*, 136:189–208, 1990.

Erich Kaltofen, Wen-Shin Lee, and Austin A. Lobo. Early termination in Ben-Or/Tiwari sparse interpolation and a hybrid of Zippel's algorithm. In [Traverso, 2000], pages 192–201.

Erich Kaltofen and B. David Saunders. On Wiedemann's method of solving sparse linear systems. In *Applied Algebra, Algebraic Algorithms and Error–Correcting Codes (AAECC '91)*, volume 539 of *Lecture Notes in Computer Science*, pages 29–38, October 1991.

Wolfgang W. Küchlin, editor. *ISSAC'97. Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation, Maui, Hawaii.* ACM Press, New York, July 1997.

Hendrik Lenstra. Factoring integers with elliptic curves. *The Annals of Mathematics, second series*, 126(3):649–673, November 1987.

Robert H. Lewis. *Fermat, a computer algebra system for polynomial and matrix computations*, 1997. http://www.bway.net/~lewis.

Jean-Pierre Massias and Guy Robin. Bornes effectives pour certaines fonctions concernant les nombres premiers. *Journal de Théorie des Nombres de Bordeaux*, 8:213–238, 1996.

Maurice Mignotte. *Mathématiques pour le calcul formel.* Presses Universitaires Françaises, 1989.

Thomas Mulders and Arne Storjohann. Diophantine linear system solving. In *International Symposium on Symbolic and Algebraic Computation (ISSAC 99)*, pages 181–188, Vancouver, BC, Canada, July 1999.

James R. Munkres. *Elements of algebraic topology*, chapter The computability of homology groups, pages 53–61. Advanced Book Program. The Benjamin/Cummings Publishing Company, Inc., 1994.

Patrick Ozello. *Calcul exact des formes de Jordan et de Frobenius d'une matrice.* PhD thesis, Université scientifique technologique et médicale de Grenoble, January 1987.

Victor J. Rayward-Smith. On computing the Smith normal form of an integer matrix. *ACM Transactions on Mathematical Software*, 5(4):451–456, December 1979.

James A. Reeds and Neil J. A. Sloane. Shift-register synthesis (modulo m). *SIAM Journal on Computing*, 14(3):505–513, August 1985.

Ernest Sibert, Harold F. Mattson, and Paul Jackson. Finite Field Arithmetic Using the Connection Machine. In Richard Zippel, editor, *Computer algebra and parallelism, Proceedings of the second International Workshop on Parallel Algebraic Computation*, volume 584 of *LNCS*, pages 51–61. Springer Verlag, May 1990.

A. Storjohann. *Algorithms for Matrix Canonical Forms.* PhD thesis, Institut für Wissenschaftliches Rechnen, ETH-Zentrum, Zürich, Switzerland, November 2000.

Arne Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. In Yagati N. Lakshman, editor, *ISSAC '96: Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation, July 24–26, 1996, Zurich, Switzerland*, pages 267–274, New York, NY 10036, USA, 1996. ACM Press.

Olga Taussky. Bounds for characteristic roots of matrices. *Duke Mathematical Journal*, 15: 1043–1044, 1948.

Carlo Traverso, editor. *ISSAC'2000: Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation, Saint Andrews, Scotland.* ACM Press, New York, August 2000.

Richard S. Varga. *Matrix iterative analysis.* Number 27 in Springer series in Computational Mathematics. Springer-Verlag, second edition, 2000.

Gilles Villard. Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems. In [Küchlin, 1997], pages 32–39.

Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, January 1986.

Richard Zippel. *Effective Polynomial Computation*, chapter Zero Equivalence Testing, pages 189–206. Kluwer Academic Publishers, 1993.