Integer Smith Form via the Valence: Experience with Large Sparse Matrices from Homology

J-G. Dumas
Unité Informatique et Distribution, B.P. 53 X, 38041 Grenoble Cedex, France.
Jean-Guillaume.Dumas@imag.fr, www.apache.imag.fr/~jgdumas

B. D. Saunders
Department of Computer and Information Sciences, University of Delaware, Newark, Delaware 19716.
saunders@udel.edu, www.eecis.udel.edu/~saunders

G. Villard
Laboratoire de Modélisation et Calcul, IMAG, BP 53 F, 38041 Grenoble cedex 9, France.
Gilles.Villard@imag.fr, www-lmc.imag.fr/lmc-cf/Gilles.Villard

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.

Keywords

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 [15, 27]. There exists also some theoretical advances for sparse matrices with iterative methods [13] but those are 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 oval 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 AAᵀ. 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 [4, 1]. 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 [24]. In the cases we studied, the boundary maps can be very large, e.g. around 10⁵ rows and columns, and very sparse, e.g. 6 entries per row.

We present an overview of our algorithm in §2; we then discuss some implementation details in §3 and §4. Finally we report on experiments using matrices from homology in §5.

2. VALENCE BASED SMITH FORM ALGORITHM
In this section we describe a new 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 asymptotic complexity is not better than Giesbrecht’s algorithm [13, Theorem 2.5]. The method is particularly effective when the degree of the minimal polynomial of AAᵀ is small. We begin with some definitions:

DEFINITIONS 1. The valence of a polynomial is its trailing non-zero 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 ≤ i ≤ min(m, n), the i-th determinantal divisor of a matrix A, dᵢ(A), is the greatest common divisor of the i × i minors of A. Let d₀(A) = 1.
The $i$-th invariant factor of $A$ is $s_i(A) = d_i(A)/d_{i-1}(A)$, or $0$ if $d_i(A) = 0$.

The Smith form of $A$ is the diagonal matrix

\[ S = \text{diag}(s_1(A), \ldots, s_{\min(m,n)}(A)) \]

of the same shape as $A$.

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 $\mathbb{Z}_q^*$.

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.

**Definition 2.** For a positive integer $q$, we define $\text{rank}_q(A)$, the rank of $A$ mod $q$, to be the greatest $i$ such that $q$ does not divide the $i$-th invariant factor of $A$, and 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 $A \in \mathbb{Z}^{n \times m}$. A may be a “black box” meaning that the only requirement is that left and right matrix-vector products may be computed: $x \to Ax$ for $x \in \mathbb{Z}^m$, $y \to yA$ for $y' \in \mathbb{Z}^n$.

**Output:** $S = \text{diag}(s_1, \ldots, s_{\min(n,m)})$, the integer Smith form of $A$.

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

2. **[Integer factorization]**
   Factor the valence $v$.
   Let $L$ be the list of primes which divide $v$.

3. **[Rank and first estimate of Smith form.]**
   Choose a prime $p$ not in $L$ (i.e. $p \nmid v$).
   Compute $r = \text{rank}_p(A)$.
   [we know now the first $r$ invariant factors are nonzero and the last $N - r$ are $0$’s.]
   Set $S = \text{diag}(s_1, \ldots, s_N)$, where $s_i = 1$ for $i \leq r$ and $s_i = 0$ for $i > r$.

4. **[Non-trivial invariant factors]**
   For each prime $p \in L$
   Compute $S_p = \text{diag}(s_{p,1}, \ldots, s_{p,N})$, the Smith form of $A$ over the local ring $\mathbb{Z}_p$. [See section 4]
   Set $S = S \otimes S_p$. [That is, set $s_i = s_{p,i}$ for those $s_{p,i}$ which are non-trivial powers of $p$.]

5. **[Return invariants]**
   Return $S = \text{diag}(s_1, \ldots, s_N) \in \mathbb{Z}^{n \times m}$.

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

**Theorem 1.** Let $A$ be a matrix in $\mathbb{Z}^{n \times m}$. Let $(s_1, \ldots, s_r)$ be its non-zero invariant factors. If a prime $p \in \mathbb{Z}$ divides some non-zero $s_i$, then $p^r$ divides the characteristic valuation of $AA^t$ and $p$ divides the minimal valuation $(AA^t)$. The same is true of the valences of $A^t$ as well.

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

Let $v_i = \text{valence}(C)$ be the characteristic valuation of $B$, and let $v_i = \text{valence}(F_i)$. It follows easily from the nature of polynomial arithmetic that $v_1|v_2|\cdots|v_r = v$, and $v_i = \prod v_i$. Hence all primes that occur in $v_i$ 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 $v_i$.

**Notations 1.** As in [17], we denote by $S_p^n$ the set of all length $i$ subsequences of $[1,n]$ and by $A_{ij}$, $I \subseteq S_p^n$, $J \subseteq S_p^n$, the $i \times i$ determinant of the submatrix of $A$ in the rows $I$ and columns $J$.

Using the Cauchy-Binet formula [12, Proposition 1.3.2.14], note that $v_i$, as a coefficient of the characteristic polynomial, satisfies for some $i$:

\[ \pm v_i = \sum_{I \subseteq S_p^n} B_{IJ} \sum_{K \subseteq S_p^n} A_{IK}A_{KJ} = \sum_{K \subseteq S_p^n} \sum_{K \subseteq S_p^n} A_{IK}^2. \]

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

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 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 (section 3 is devoted to details on part 1 of the Valence algorithm and section 4 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. In this section we propose two different ways to compute such a set. One method is by computing the valence, the other is a partial integer elimination.
3.1 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.

3.1.1 Degree of the minimal polynomial

To compute the degree of the integer minimal polynomial, 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. Primes may have a lower degree minimal polynomial. We call them "bad" primes. Let \( d \) be the degree of the integer minimal polynomial. There exists a vector \( u \) such that the Krylov subspace associated to \( B \) and \( u \), \( \text{span}(u, Bu, \ldots, B^d u) \), is of rank \( d \). Therefore there exists a square \( d \times d \) non-zero minor, \( M_u \), of the matrix \( [u, Bu, \ldots, B^d u] \). "Bad" primes must divide this minor. Given an upper bound on \( M_u \) we can then give an upper bound \( U \) on the number of bad primes. Let \( \beta \) be an upper bound to the norm of the rows of \( B \). Next, using Hadamard’s inequality [31, Theorem 16.6], we can state that \( |M_u| \leq U \), where \( U = 2^d \beta^2 \). Suppose we choose primes at random from a set \( P \) of primes greater than a lower bound \( l \). There can be no more than \( \log(U) \) primes greater than \( l \) dividing \( M_u \). 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 \( n \)-th prime, \( p_n \) [21, Theorem A]:

\[
\begin{align*}
n(\log(n) + \log(\log(n))) - 1.002872 & \leq p_n & n \geq 2 \\
p_n \leq n((\log(n) + \log(\log(n))) - 1 + 1.8 \frac{\log(\log(n))}{\log(n)} ) & n \geq 13
\end{align*}
\]

Now, for random primes, each polynomial modulo a prime has a probability at least \( 1 - \frac{1}{\log(p)^k} \) of being of degree \( d \). Suppose that there are \( k \) polynomials of maximal degree. The probability that this degree is \( d \) is then at least \( 1 - (\frac{1}{\log(p)^k})^k \).

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 gives us only 1.7% of bad primes. With only 10 polynomials this reduces to a probability of failure less than \( 2 \times 10^{-16} \).

The next question is how many primes must be used for the Chinese remaindering. Using Hadamard’s inequality again would induce a use of \( O(n) \) minimal polynomial computations. 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 parallel computation, it is interesting to have a sharper estimate. We can use the ovals of Cassini to bound the spectral radius and hence the valence.

3.1.2 Early termination

In our computations for which timings and results are reported here, we compute \( v_p \), the minimal valence modulo \( p \), 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 we will see in section 3.1.3. But when both these bounds are large, we use a probabilistic termination condition. Let \( v_k \) be \( v \) reduced mod \( M = \prod_{i=1}^k p_i \) for randomly chosen primes \( p_i \). 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 (this can happen 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 modulo \( p^* \). Compute also \( v_k \mod p^* = v_k \), then if \( v_k = v \), the two valence modulos \( p^* \) 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 \( v_k = v \) with high probability \( v \). We make this probability explicit in the following lemma.

**Lemma 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 \( \{p_1, \ldots, p_k\} \) 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 \( 1 - \frac{\log(\frac{p_k}{p^*})}{p^*} \).

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

For instance, the worst example given in section 3.1 is a matrix for which the valence is bounded by \( U = 117^{247} \). We choose some primes greater than \( l = 2^{15} \). We can suppose that \( M \), the product of primes, is greater than \( 2 \), therefore \( \log_2(\frac{U}{M}) \leq 379 \). 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 bound is closer to \( 10^{200} \). There, only one application of the trick gives more than 98.5% confidence!

3.1.3 Ovals of Cassini

For a parallel computation of the valence, in particular, a sharp bound is very useful. The Hadamard bound seems too pessimistic an estimation for sparse matrices. Therefore, we use a bound determined by consideration of Gershgorin disks and ovals of Cassini. This bound is of the form \( \beta^d \) where \( \beta \) is a bound on the eigenvalues and \( d \) is the degree of the minimal polynomial. Thus \( n \) does not appear as an exponent as in the Hadamard bound. To compute the degree we compute several polynomials and take the maximal degree as described in the beginning of this section. To compute
a bound on the eigenvalues we use Gershgorin disks and Cassini ovals.

The i-th Gershgorin disk is centered at \( a_{ii} \) and has for a radius the sum of the absolute values of the other entries on the i-th row. Gershgorin’s theorem is that all of the eigenvalues are contained in the union of the Gershgorin disks [28]. One can then go further and consider the ovals of Cassini [5, 6], which may produce sharper bounds. For our purposes here it suffices to note that each Cassini oval is a subset of two Gershgorin circles, and that all of the eigenvalues are contained in the union of the ovals. We can then use the following proposition bounding the coefficients of the minimal polynomial:

**Proposition 1.** Let \( B \in \mathbb{R}^{n \times n} \) with its spectral radius bounded by \( \beta \). Let \( M(X) = \minpoly(B) = \sum_{k=0} \lambda_k X^k \). Then \( \text{valence}(B) \leq \beta^d \) and \( \forall i \in [0..d], |m_i| \leq \max(\sqrt{\beta^3}, \beta)^d \).

**Proof.** It suffices to note that \( |m_i| \leq (\beta^d)^d \) [22, Theorem IV.4.1].

When \( d \) is small relative to \( n \) this is an improvement on the Hadamard bound since the latter would be of order \( n \) rather than \( d \).

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 (e.g. the matrix \( AA^t \) is diagonally dominant).

There remains to compute a 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 operations and two radii of such an oval. Then any point \( \lambda \) of this oval satisfies the following: \( |\lambda - q_i| |\lambda - q_j| < r_i r_j \) [5, Theorem 1]. We want to know the maximal absolute value of such a \( \lambda \). Set \( \delta = q_i - q_j > 0 \). First if \( \lambda > q_i \), set \( X = \lambda - q_i \). Then \( X \) satisfies \( X^2 + \delta X - r_i r_j < 0 \) and \( X > 0 \). Hence \( |X| < \sqrt{\delta r_i r_j} \). Next if \( q_i > \lambda > q_j \), then \( |X| < q_i < q_j \). Last if \( q_j > \lambda \), set \( X = \lambda - q_j \). Then \( X \) satisfies \( X^2 + \delta X - r_i r_j < 0 \) and \( X < 0 \). Hence \( |X| < \sqrt{\delta r_i r_j} \). Therefore \( \beta_{OC} \) as in the algorithm matches the requirements. The complexity analysis is straightforward.

### 3.2 A Point about the Symmetrization

The choice between \( AA^t \) and \( AA^t \) is easily made in view of the following.

**Theorem 3.** \( \minpoly(AA^t) \) and \( \minpoly(AA^t) \) are equal or differ by a factor of \( x \).

**Proof.** Let \( m_{AA^t}(x) \) and \( m_{AA^t}(x) \) be respectively the minimal polynomials of \( AA^t \) and \( AA^t \). The Cayley-Hamilton theorem [12, Theorem IV.4.2] states that \( m_{AA^t}(AA^t) = 0 \). Then \( \text{Am}_{AA^t}(AA^t) = 0 \) and \( (Xm_{AA^t})(AA^t) = 0 \). Since \( m_{AA^t} \) is the minimum polynomial of \( AA^t \) it follows that \( m_{AA^t} | Xm_{AA^t} \). We can similarly prove that \( m_{AA^t} | Xm_{AA^t} \). Then either \( m_{AA^t} = Xm_{AA^t} \) or \( m_{AA^t} = Xm_{AA^t} \) or \( Xm_{AA^t} = m_{AA^t} \).

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 \( AA^t \) in the algorithm, to reduce the cost of the inner products involved. Moreover any bound on the coefficients of \( \minpoly(AA^t) \) can then be applied to those of \( \minpoly(AA^t) \) and vice versa.

### 3.3 An Alternative to the Valence

Finally, we mention that, in some experiments, we also have used integer elimination to compute a multiple of the last non-zero invariant factor. Here we have used essentially Storjohann’s Bareiss-based probabilistic Smith form algorithm [27], abbreviated. We compute an integer Bareiss-LU decomposition of the matrix storing only some parts of \( U \). The last non-zero row of this matrix, say the \( i \)-th row, contains only \( i \times i \) minors [2, 19]. Therefore the last non-zero invariant factor divides the gcd of the last non-zero row of \( U \). We were able to succeed with this heuristic on some sparse matrices, where complete execution of Storjohann’s algorithm would have required too much memory. Moreover, as this algorithm is deterministic, the gcd computed is a true multiple of the invariant factor.

### 4. Local Smith Form at P

Next consider the question of computing the local Smith form in \( \mathbb{Z}^k \). 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.

4.1 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 $p^r$, which determines the ranks mod $p^k$ for $k < r$ and hence the powers of $p$ in the Smith form up to $p^{r-1}$. Suppose by this means we find that $s_r$ is not zero mod $p^r$, 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 $p^r$ is less than $r$, we can compute directly in $\mathbb{Z}[p]/p^r$. Due to intermediate expression swell, it is not effective to repeat the LRE computation with larger exponent $e$ until $s_r$ is nonzero mod $p^r$.

**Algorithm: LRE [Local-Ranks-by-Elimination-mod-$p^r$]**

**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:**
- $r_{p^r}$, for $1 \leq i \leq e$.

1. [Initializations]
   - set $k = e$
   - set $r = 0$

2. [e successive Gauss steps]
   - for (exponent $k = 1$ to $e$)
     - while ($\exists (s, t) \in [r..m] \times [r..n], p | a_{st}$)
       - Swap rows $r$, $s$, and columns $r$, $t$
       - for all $(i, j) \in [r+1..m] \times [r+1..n]$
         - if ($i, j$) is the pivot
           - elimination, with division, mod $p^{e-k+1}$
           - set $a_{ij} = a_{ij} - a_{ik}a_{kj}/a_{kk} (mod\ p^{e-k+1})$
           - set $r = r + 1$
           - set $p_{r, p^e} = r$
           - and invariant factors $s_i = p^{e-k-1}$ for $r_{p^e-1} < i \leq r_{p^e}$
     - for all $(i, j) \in [r..m] \times [r..n]$
       - set $a_{ij} = a_{ij}/p$

3. [Return local ranks]
   - return $r_{p^e}$, for $1 \leq i \leq e$

**Theorem 4.** For a positive integer $e$, algorithm Local Ranks by Elimination mod $p^e$ is correct and runs using $O(r\ m\ n)$ arithmetic operations 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 $D_k = diag(1, \ldots, 1, p, \ldots, p)$, with $r_{p^k}$'s $1$'s. Then in effect, $A$ has been factored as $P \prod_{p=1}^r L_k D_k' A' Q'$, 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 $D_k A' = D_k U$, 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 $B = \prod_{p=1}^r L_k D_k$. From this it is easily seen that the the Smith form of $A$ in $\mathbb{Z}[p^e]$ is $S = \text{diag}(s_r) = \prod_{p=1}^r D_k$, because for both $B$ and $S$ and for each $j$, the leading principal $j \times j$ minor contains the the least power of $p$, namely $\prod p_i^{s_i}$. To see this for $B$, use Cauchy-Binet expansion of the minors together with the lower triangularity of the $L_k$. □

4.2 Iterative Methods

4.2.1 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. This section proposes three iterative methods. The first one is a quick check that the integer rank is the rank mod $p$. The second one is to use Wiedemann’s algorithm [32] with diagonal scaling [11, Theorem 6.2] to compute the rank mod $p$. This algorithm has much lower memory requirements than elimination, has better asymptotic complexity and is effective in practice for actually large sparse matrices over large fields [8]. Still it doesn’t give the complete local Smith form at $p$. In 4.2.3 we propose a $p$-adic way to compute the last invariant factor of this local Smith form at $p$, completing the algorithm.

4.2.2 Null Space Method

Consider a prime $p$ which occurs in the Smith Form of $A$. We know that $p^e$ divides the characteristic valence of $AA^t$. It seems more likely in general that $p^e$ 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 $A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$; it has Smith form $[1, 0]$ . However $AA^t = A^t A = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ and has minimum 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 these 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 a 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 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 $R(B)$. 

5
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 BlackBox.
- the minimal polynomial $M$ of $AA^T$, and a factor $R$ of $M$.
- $\varepsilon \in \mathbb{R}$ such that $0 < \varepsilon < 1$.

**Output:**
- a list $\mathcal{T}$ 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 $1 - \varepsilon$.

1. [Initializations]
   - Set $\mathcal{T} = \emptyset$
   - Set $d = \deg(R)$.
   - Let $N = \frac{M}{R}$.
   - Set $g = \lfloor \varepsilon^{-1} \rfloor$.
   - Set $p_0$ a random prime such that $p_0 > g$ and $p_0 \nmid \valence(M)$.
   - Form the BlackBox $N(B)$.

2. [Probabilistic Null-Space dimension]
   - Select $d + 1$ random nonzero vectors $u_i \in (\mathbb{Z}_p)^n$.
   - For each $1 \leq i \leq d$, set $v_i = N(B)u_i$.
   - Let $N' = N(B)$, $d' = \dim_k N(B)$.
   - For each prime $p$ dividing $\valence(R)$
     - if $p^2$ does not divide $\valence(M)$, add $p$ to $\mathcal{T}$.

3. [Return not occurring primes]
   - return $\mathcal{T}$.

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

**Lemma 2.** Let $B \in \mathbb{Z}^{n \times n}$. Let $N, R \in \mathbb{Z}[X]$ 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, using [12, Theorem VII.2.1], we know that $\ker(R(B))$ and $\ker(N(B))$ are supplementary. We conclude by the dimension theorem, proving the lemma.

**Theorem 5.** Algorithm Null-Space-Dimension is correct.

**Proof.** Let $B = AA^T$. $B$ 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 $P$. 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 $B$. 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 $(\mathbb{Z}_p)^n$.

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 $\mathbb{Z}_p$ with $h$ elements. Then $P(j, n) = P(j - 1, n) + P$ (first $j - 1$ independent but $j$-th dependent) which gives $P(j, n) \leq P(j - 1, n) + \frac{h^{j-1}}{h^n} \leq \frac{h^{j-1}}{(n - 1)!h^n} < (h - 1)^{j-1}/n$. Hence, as $h > g$, if $k \geq 2$ then $P(d + 1, kd) \leq g^{d-kd} \leq \frac{1}{\varepsilon}$.

**4.2.3 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 $p^r$. The method of Reeds and Sloan [25] can be adapted to compute the annihilator of our matrix $Z_p$. It may be possible to adapt this to the purpose of computing the rank of the matrix in $Z_p$. 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 non-zero 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 $\kappa = k_0$ of $p$ in the last nonzero invariant factor. Since determining whether $\kappa$ 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 $\kappa$ rather than in $\Sigma \kappa_i$, 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 $\text{ord}_p(x)$ 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 $\text{ord}_p(x) = \kappa$. To reduce the cost of computing $\kappa$, 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 $v_1$, $1 \leq i \leq n$, and let $u$ be the first canonical vector. Define the $n \times n$ matrix $E(v)$ by:

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

**Lemma 3.** Let $b$ and $v$ be two random vector integer entries with entries chosen uniformly in $[0, p - 1]$. With probability $(1 - 1/p)^I$, $E(v) + wv^T$ is invertible ($v \neq 0$) and $\kappa$ is the order of the first entry of the solution $y$ to $AE(v) + wv^T - y = b$. When $v \neq 0$, the order cannot be strictly greater than $\kappa$.

**Proof.** If $E(v) + wv^T$ is invertible and $z$ denotes the solution to $Az = b$ then $y = (E(v) + wv^T)x$ and the first entry of $y$ is $y_1 = \Sigma_i v_i z_i$. Let $U$ and $V$ be unimodular transformations such that $S = UAV$ is in Smith form, $U$ and $V$ define
bijections on \([0, p - 1]^{n}\) and on \((\mathbb{Z}/p\mathbb{Z})^{n}\) thus:
\[
\mathcal{P}_b = \mathcal{P}_{\text{order}}(\text{ord}_b(x) = \kappa; Ax = b) = \mathcal{P}_{\text{order}}(\text{ord}_b(x) = \kappa; S(V^{-1})x = Ub) = \mathcal{P}_{\text{order}}(\text{ord}_b(z) = \kappa; Sz = b) \geq \mathcal{P}_{\text{order}}(1 \leq b_0 \leq p - 1) = 1 - 1/p.
\]

If \(\text{ord}_b(x) = \kappa\) then \(p^n x \not\equiv 0 \mod p\). With probability \(1 - 1/p\) we have also \(p^n \sum_i v_i x_i \not\equiv 0 \mod p\) and the order of \(y_t\) is \(\kappa\). In this latter case \(v\) is nonzero and by construction \(E(v) = \pm 1\) is invertible. This proves the first assertion of the lemma. For the last assertion it suffices to notice that the \(v_i\)'s are integers, the order of \(\sum_i v_i x_i\) is lower than the order of \(x\).

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

**Lemma 4.** 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 \(\tau\) are the same.

**Proof.** The matrices \(A\) and \(\tilde{A}\) as matrices in \((\mathbb{Z}/(p^{\tau}))^{n \times n}\) are equivalent.

**Lemma 5.** Let \(b\) be an integer vector with \(||b||_{\infty} \leq ||A||_{\infty} = \beta\). Any entry of the solution \(x\) to \(Ax = b\) may be computed in \(N = \tau (\log n + 2 \log_2 (p^\tau + q\beta)) + 2\) multiplications of \(A\) times a vector with entries bounded by \(q\) and \(O(n^2 \log_2 n + 2 \log_2 (p^\tau + q\beta))^2\) additional binary operations. In addition to the matrix storage, the algorithm requires an \(O(n \log n + 2 \log_2 (p^\tau + q\beta))\) bits of storage.

**Proof.** We apply the algorithm of Dixon [7] based on a q-adic expansion of the solution. The matrix \(\tilde{A}\) is invertible in \(\mathbb{Q}^{n \times n}\) and in \((\mathbb{Z}/p\mathbb{Z})^{n \times n}\), \(\tilde{A}^{-1} \equiv p^{-\tau} I d\ mod q\). The number \(N\) of iterations is given in the lemma, each iteration has binary cost \(O(N \log_2 q)\) when the q-adic expansion of only one entry of the solution is computed. The rational value of that entry is constructed from the expansion within the same cost. The extra amount of storage needed is \(O(N \log_2 q)\).

For any integer \(\tau\), from lemma 3 and lemma 4 we may give a randomized algorithm which compares \(\tau\) to \(\kappa\) with an arbitrary error tolerance \(\varepsilon > 0\). By repeated random choices of \(q, b, v\) it is possible to produce an algorithm returning, with probability as low as required, \(\kappa\) if \(\kappa \leq \tau\) (maybe wrong) and reporting that \(\kappa\) is strictly greater than \(\tau\) otherwise (always correct).

**Algorithm: LIF [Largest Invariant Factor] order**

**Input:**
- \(A \in \mathbb{Z}^{m \times n}\) invertible,
- a prime \(p\), an integer \(\tau \geq 1\),
- an error tolerance \(\varepsilon > 0\).

**Output:**
- with probability at least \(1 - \varepsilon\), 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 \(p \neq p\).
Build \(\tilde{A} = p^{\tau+1} + qA\).

(2) **[Order]**
For \(t\) from 1 to \([\log(\varepsilon)/\log(1 - (1 - 1/p)^2)]\) do
- Choose random \(b\) and \(v\) in \([0, p - 1]^n\).
- If \(v\) \(\not\equiv 0\) construct \(E(v) + \text{gcd}(E(v), q)\).
- Compute \(y_t \in \mathbb{Q}\) the first entry of \(y\) such that \(A(E(v) + \text{gcd}(E(v), q))^{-1}y = b\).
- If \(\text{ord}_p(y_t) > \tau\) then Return(\("\kappa > \tau\"\)).
- max order := max\{max order, \text{ord}_p(y_t)\}.

Return(\("\kappa = \text{max order}\"").

**Proposition 2.** The algorithm LIF works as specified, if \(\kappa \leq \tau\) then it returns \(\kappa\) with probability at least \(1 - \varepsilon\) or a lower value with probability less than \(\varepsilon\). If \(\kappa > \tau\) then it discovers the latter inequality with probability at least \(1 - \varepsilon\), this result is always correct, or returns a wrong value lower than \(\tau\) with probability less than \(\varepsilon\). The cost of the algorithm is bounded by the cost in lemma 5 times the number of passing through the for loop (a function of the error tolerance and of \(p\)).

**Proof.** If \(\kappa \leq \tau\) then by lemma 4 all the invariant factors of \(A = p^{\tau+1} + qA\) are of order less than \(\kappa\). By lemma 3, the probability that \(v = 0\) or that \(y_t\) gives an order strictly less than \(\kappa\) is \(1 - (1 - 1/p)^\tau\). The computed order cannot be strictly greater than \(\kappa\). After \([\log(\varepsilon)/\log(1 - (1 - 1/p)^\tau)]\) trials, the probability to have a wrong result is less than \(\varepsilon\). If \(\kappa > \tau\), the largest invariant factor of \(\tilde{A}\) is at least of order \(\tau + 1\), in the same way, this will be found with probability at least \(1 - \varepsilon\). This latter result is always correct since by lemma 3 the computed order cannot be greater than the actual one. The cost bounds are immediately derived (multiplying by the inverse of \(E(v) + \text{gcd}(E(v), q)\) requires linear time only).

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 [14, 55] for Diophantine equations, the number of trials could be reduced up to some increase in time for the applications of lemma 5.

- **General case.** When \(A\) is \(m \times n\) singular of rank \(r\), unlike in lemma 4, the use of the modified matrix \(\tilde{A}\) will always introduce new nonzero invariant factors that prevent from computing \(\kappa\) 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 \(S_I = \text{diag}(p^{s_1}, p^{s_2}, \ldots, p^{s_r})\). Such a matrix \(A_S\) may be obtained by conditioning \(A\). We generalize the construction of [23, Lemma 11]. Let us denote by \(U\) and \(V\) two unimodular multipliers such that \(UAV\) is in Smith form:
- \(UAV = [S_{ij}]\) and consider two pre-conditioners \(P \in \mathbb{Z}^{r \times m}\) and \(Q \in \mathbb{Z}^{n \times r}\). If \(T \in \mathbb{Z}^{n \times r}\) is the matrix constructed with the first \(r\) columns of \(U^{-1}\) and if \(W \in \mathbb{Z}^{n \times n}\) is constructed with the first \(r\) rows of \(V^{-1}\) then: \(A = TS\bar{U}W\) 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 6. There exists two matrices $W \in \mathbb{Z}^{r \times n}$, $T \in \mathbb{Z}^{m \times r}$ only depending on $A$ giving: if $P \in \mathbb{Z}^{m \times n}$ and $Q \in \mathbb{Z}^{n \times r}$ are such that $p \det(PT)$ and $p \det(WQ)$ then $A_S = PAQ$ is invertible and its local Smith form at $p$ is the invertible submatrix $S$ of the local Smith form of $A$.

For pre-conditioners satisfying the lemma condition one may consider Toeplitz matrices as used in [18, Theorem 2] or in [14, §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 $\eta$ and with coefficients bounded in absolute value by $p$. Let $\theta \equiv x \mod \Gamma$ and $V_p$ be the set of the polynomials of degree less than $\eta$ with coefficients 0 or 1 over $\mathbb{Z}[\theta]$.

Proposition 3. [14] Let $A \in \mathbb{Z}^{m \times n}$ be of rank $r$, $p$ a prime. Consider as pre-conditioners $\mathcal{P}$ and $\mathcal{Q}$, an upper and lower triangular Toeplitz matrix with unit diagonal and whose other entries are randomly chosen in $V_p$. The local Smith form of $A_S = \mathcal{P}AQ$ at $p$ is $S$ 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 [14], if $y_1 = \sum_{0 \leq j \leq q} y_1(\theta)^j \in \mathbb{Q}(\theta)$ is the first entry of the solution to $A_S(E(v) + w) y = b$, one looks at the order of $y_1(\theta)$. The sizes of the integers (numerators and denominators) involved in the computations are a factor $O(\eta)$ i.e. $O(\log r)$ greater than for the regular case. The cost bounds of lemma 5 will be modified by the same factor.

Instead of Toeplitz matrices one may use sparse matrices as in [32, §III] or [30] to avoid order fields.

Proposition 4. [32, Theorem 1'] Let $A \in \mathbb{Z}^{m \times n}$ be of rank $r$, $p$ a prime. A random procedure exists for generating sparse pre-conditioners $\mathcal{P}$ and $\mathcal{Q}$ with coefficients in $\{0, 1\}$ and of total Hamming weight $O(r \log r)$. With probability at least $1 - \epsilon$ they satisfy that the local Smith form of $A_S = \mathcal{P}AQ$ at $p$ is $S$.

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 $O(\log r)$.

5. 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 [24]. We will mainly talk about three homology matrix classes. The matrices will be denoted by three naming patterns:

- **mki.bj** denotes the boundary matrix $j$ from the matching complex with i vertices.
- **chi-k.bj** denotes the boundary matrix $j$ from the i by k chessboard complex.
- **nick.bj** denotes the boundary matrix $j$ from the not i-connected graph with k vertices.

For details on those simplicial complexes see [1, 3].

The boundary matrices are sparse matrices with a fixed number $k$ of non-zero 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 $A_i A_i'$ also have $-1, 0$ and $1$ as entries except for the diagonal which is respectively all $l$ and all $l$. However, as expected, those Laplacians have more than twice as many non-zero elements as $A_i$. Thus, we did not perform the matrix multiplications to compute $A_i A_i' v$. We performed two matrix-vector products, $A_i (A_i' v)$, instead.

We will also check in section 5.1 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 1 illustrates the patterns occurring in these matrices. It shows ch4.b2, the boundary map between the second and first dimensions of the 4 x 4 chessboard complex, together with its Laplacians. It is of size 96 x 72 with 288 non-zero elements. It has 3 elements per row and 4 per column. On the other hand $A_i A_i'$ is of size 96 x 96 with 960 non-zero elements and $A_i A_i'$ of size 72 x 72 with 648 non-zero elements.

Our experiments were realized on a cluster of 20 Sun Microsystems 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 [26]. The algorithms were implemented in $C$++ with the LINBOX library for computer algebra and the ATHAPASCAN\textsuperscript{2} environment for parallelism.

5.1 Experiments

We will call $\omega$ the number of non-zero elements per row, $s$ the shape and $r$ the integer rank of the matrix under consideration. We will produce these elements only in table 1. 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 1 we present computations of the integer minimum 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 minimum polynomial, $d_m$ 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 another composite factor of 376 digits that we were unable to factor. 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,\footnote{Symbolic linear algebra library, http://www.cis.udel.edu/~caviness/linbox}

\footnote{Parallel execution support of the APACHE Project, http://www.apache.imag.fr/software}
Figure 1: Chessboard complex 4-4, boundary matrix 2 with $AA^t$ and $A^tA$.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$\omega_s, r$</th>
<th>CB</th>
<th>Rem</th>
<th>$d_m$</th>
<th>$v_m$</th>
<th>Sequential</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>mk9.b3</td>
<td>4, 945x1260, 875</td>
<td>12</td>
<td>2</td>
<td>$X^4$</td>
<td>$2^3, 3^2$</td>
<td>0.31</td>
<td>0.39</td>
</tr>
<tr>
<td>mk10.b3</td>
<td>4,4725x3150,2564,</td>
<td>24</td>
<td>3</td>
<td>$X^7$</td>
<td>$2^3, 5^3, 7, 13$</td>
<td>1.16</td>
<td>0.81</td>
</tr>
<tr>
<td>mk11.b4</td>
<td>5, 9450x17325, 9324</td>
<td>14</td>
<td>3</td>
<td>$X^9$</td>
<td>$-2^3, 3^2, 5^2, 7, 11$</td>
<td>4.00</td>
<td>1.80</td>
</tr>
<tr>
<td>mk12.b3</td>
<td>4, 51975x13860, 12440</td>
<td>60</td>
<td>3</td>
<td>$X^7$</td>
<td>$2^{10}, 3^2, 5, 7, 11, 13$</td>
<td>16.68</td>
<td>5.55</td>
</tr>
<tr>
<td>mk12.b4</td>
<td>5, 63270x51975, 39535</td>
<td>30</td>
<td>4</td>
<td>$X^{11}$</td>
<td>$-2^{13}, 3^2, 5^2, 7, 11$</td>
<td>34.49</td>
<td>11.39</td>
</tr>
<tr>
<td>mk13.b4</td>
<td>5, 270270x135135, 111463</td>
<td>50</td>
<td>4</td>
<td>$X^{11}$</td>
<td>$-23^{11}, 3^7, 5^2, 7, 11, 13$</td>
<td>149.06</td>
<td>42.13</td>
</tr>
<tr>
<td>mk13.b5</td>
<td>6, 135135x270270, 134211</td>
<td>18</td>
<td>4</td>
<td>$X^{12}$</td>
<td>$2^{10}, 3^2, 5, 7^2, 11, 13$</td>
<td>96.11</td>
<td>29.41</td>
</tr>
<tr>
<td>ch4-4.b2</td>
<td>3, 96x72, 57</td>
<td>12</td>
<td>1</td>
<td>$X^4$</td>
<td>$2^7, 3$</td>
<td>0.08</td>
<td>0.24</td>
</tr>
<tr>
<td>ch5-5.b3</td>
<td>4, 600x600, 424</td>
<td>16</td>
<td>3</td>
<td>$X^4$</td>
<td>$2^3, 5, 7^2$</td>
<td>0.35</td>
<td>0.39</td>
</tr>
<tr>
<td>ch6-6.b4</td>
<td>5, 4320x5400, 3390</td>
<td>20</td>
<td>3</td>
<td>$X^{10}$</td>
<td>$2^{10}, 3^2, 5^2, 7$</td>
<td>1.83</td>
<td>1.43</td>
</tr>
<tr>
<td>ch7-7.b4</td>
<td>5, 52920x29400, 22884</td>
<td>45</td>
<td>6</td>
<td>$X^{15}$</td>
<td>$-19.17.2^{11}, 3^7, 5^2, 7, 11, 13$</td>
<td>45.17</td>
<td>10.28</td>
</tr>
<tr>
<td>ch7-7.b5</td>
<td>6, 35280x52920, 29448</td>
<td>24</td>
<td>4</td>
<td>$X^{12}$</td>
<td>$-29.41.367.34504396919539$</td>
<td>23.88</td>
<td>7.52</td>
</tr>
<tr>
<td>ch7-8.b5</td>
<td>6,141120x141120, 92059</td>
<td>36</td>
<td>7</td>
<td>$X^{21}$</td>
<td>$19.17.2^4, 13^2, 5^2, 7, 11, 13$</td>
<td>123.18</td>
<td>24.85</td>
</tr>
<tr>
<td>ch7-9.b3</td>
<td>4, 105840x17640, 16190</td>
<td>96</td>
<td>7</td>
<td>$X^{16}$</td>
<td>$29.19.2^{19}, 3^{10}, 5, 7^2, 11, 13, 31$</td>
<td>82.57</td>
<td>15.04</td>
</tr>
<tr>
<td>ch7-9.b4</td>
<td>5, 317520x105840, 89650</td>
<td>75</td>
<td>9</td>
<td>$X^{22}$</td>
<td>$-143.3^2$</td>
<td>471.22</td>
<td>64.67</td>
</tr>
<tr>
<td>ch7-9.b5</td>
<td>6, 423360x37520, 227870</td>
<td>48</td>
<td>9</td>
<td>$X^{24}$</td>
<td>$23.19.17.2^{24}, 3^{10}, 5^2, 7, 11, 12^{13} \times 2$</td>
<td>775.33</td>
<td>106.15</td>
</tr>
<tr>
<td>ch8-8.b4</td>
<td>5, 376230x17600, 100289</td>
<td>80</td>
<td>7</td>
<td>$X^{16}$</td>
<td>$29.19.17.2^{22}, 3^2, 5^2, 7, 11, 21, 13$</td>
<td>386.13</td>
<td>64.78</td>
</tr>
<tr>
<td>ch8-8.b5</td>
<td>6, 564480x376320, 276031</td>
<td>54</td>
<td>7</td>
<td>$X^{19}$</td>
<td>$-19.17.2^{12}, 3^2, 5^2, 7, 11, 21, 13$</td>
<td>732.44</td>
<td>127.57</td>
</tr>
<tr>
<td>ch8-8.b6</td>
<td>7, 322560x564180, 279237</td>
<td>28</td>
<td>5</td>
<td>$X^{14}$</td>
<td>$2^{15}, 3^4, 5, 7, 11, 13$</td>
<td>337.72</td>
<td>77.52</td>
</tr>
<tr>
<td>ch8-8.b7</td>
<td>8, 40320x322560, 40320</td>
<td>8</td>
<td>1</td>
<td>$X^1$</td>
<td>$-2^3$</td>
<td>10.63</td>
<td>10.47</td>
</tr>
<tr>
<td>ch9-9.b3</td>
<td>4, 381024x42336, 39824</td>
<td>144</td>
<td>5</td>
<td>$X^{10}$</td>
<td>$-73.6287, 5.7$</td>
<td>190.74</td>
<td>45.10</td>
</tr>
<tr>
<td>ch9-9.b4</td>
<td>5, 1905120x381024, ??</td>
<td>125</td>
<td>8</td>
<td>$X^{17}$</td>
<td>$-2^{13}, 3^{12}, 5^6, 7, 13^2, 3^2$</td>
<td>2279.32</td>
<td>538.97</td>
</tr>
<tr>
<td>n2c6.b6</td>
<td>7, 5715x9495, 2943</td>
<td>63</td>
<td>25</td>
<td>$X^{16}$</td>
<td>$17^2, 23^2, 3^2, 5^2, 7, 13^2, 3^2, C_0$</td>
<td>55.43</td>
<td>3.44</td>
</tr>
<tr>
<td>n2c6.b7</td>
<td>8, 39900x5715, 2772</td>
<td>64</td>
<td>58</td>
<td>$X^{12}$</td>
<td>$17^2, 23^2, 3^2, 5^2, 7, 13^2, C_0$</td>
<td>206.18</td>
<td>7.51</td>
</tr>
<tr>
<td>n3c6.b9</td>
<td>10, 2511x9495, 1896</td>
<td>60</td>
<td>15</td>
<td>$X^{10}$</td>
<td>$17^2, 23^2, 3^2, 5^2, 7, 13^2, C_0$</td>
<td>15.80</td>
<td>1.83</td>
</tr>
<tr>
<td>n4c5.b6</td>
<td>6, 4340x2852, 1866</td>
<td>60</td>
<td>39</td>
<td>$X^{10}$</td>
<td>$-216, 3^{10}, 5^{16}, 13^2, 191, 2^{13}, 7, 13, 29, C_0$</td>
<td>82.76</td>
<td>4.04</td>
</tr>
<tr>
<td>n4c5.b7</td>
<td>7, 4735x4340, 2474</td>
<td>63</td>
<td>82</td>
<td>$X^{16}$</td>
<td>$19.17.2^{12}, 3^{10}, 5^2, 7, 67, 2^9, C_0$</td>
<td>401.05</td>
<td>11.49</td>
</tr>
<tr>
<td>n4c5.b8</td>
<td>8, 3635x4735, 2261</td>
<td>64</td>
<td>100</td>
<td>$X^{20}$</td>
<td>$-17.2^{14}, 3^{12}, 5^2, 7, 11, 13, 31, C_0$</td>
<td>511.69</td>
<td>14.26</td>
</tr>
<tr>
<td>n4c5.b9</td>
<td>9, 1895x3635, 1374</td>
<td>63</td>
<td>62</td>
<td>$X^{14}$</td>
<td>$17^2, 2^6, 3^5, 5^2, 7, 11, 13, 31, 3^2, C_0$</td>
<td>125.93</td>
<td>5.30</td>
</tr>
<tr>
<td>n4c6.b5</td>
<td>5, 1818x320058, 13228</td>
<td>96</td>
<td>44</td>
<td>$X^{10}$</td>
<td>$-23.25, 52009, C_0$</td>
<td>1161.39</td>
<td>35.35</td>
</tr>
<tr>
<td>n4c6.b12</td>
<td>13, 25605x6923, 20165</td>
<td>117</td>
<td>358</td>
<td>$X^{20}$</td>
<td>$-2^{24}, 3^{10}, 5, 7, 11, 13, 31, 3^2, C_0$</td>
<td>66344.40</td>
<td>1181.61</td>
</tr>
<tr>
<td>n4c6.b13</td>
<td>14, 6300x25405, 5440</td>
<td>98</td>
<td>37</td>
<td>$X^{20}$</td>
<td>$-2^{21}, 3^{10}, 5, 7, 11, 13, 31, 5^2, C_0$</td>
<td>249.28</td>
<td>9.90</td>
</tr>
</tbody>
</table>

Table 1: Computing the Integer Minimal Polynomial of $AA^t$ via Chinese remaindering.
minimum polynomial over \( \mathbb{Z}_p \), and Chinese remaining of integers. The algorithm has 3 steps. First compute the Cassini bound and some minimum polynomials in parallel. Using the bound and the degree of the minimum polynomials, the maximum number of remainders needed is known. Therefore, the second step is the computation in parallel of some more minimum polynomials as required [9]. The last step is a Chinese remaining of the coefficients. A future implementation will also use parallel matrix-vector products as well as block methods [16, 29] to improve speed.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Elimination</th>
<th>Wiedemann</th>
</tr>
</thead>
<tbody>
<tr>
<td>mk9.b3</td>
<td>0.26</td>
<td>2.11</td>
</tr>
<tr>
<td>ch7-b6</td>
<td>4.67</td>
<td>119.53</td>
</tr>
<tr>
<td>ch7-b4</td>
<td>49.32</td>
<td>416.97</td>
</tr>
<tr>
<td>ch7-b5</td>
<td>2179.62</td>
<td>4283.4</td>
</tr>
<tr>
<td>ch8-8-b5</td>
<td>MT</td>
<td>55 hours</td>
</tr>
<tr>
<td>n26-b6</td>
<td>6.44</td>
<td>72.96</td>
</tr>
<tr>
<td>n26-b7</td>
<td>3.64</td>
<td>57.10</td>
</tr>
<tr>
<td>n45-b6</td>
<td>2.73</td>
<td>51.75</td>
</tr>
<tr>
<td>n46-b12</td>
<td>231.34</td>
<td>4131.06</td>
</tr>
<tr>
<td>n46-b13</td>
<td>8.92</td>
<td>288.57</td>
</tr>
</tbody>
</table>

Table 2: rank modulo 65521, Elimination vs. Wiedemann

In table 2 we report some comparisons between Wiedemann’s algorithm and elimination with reordering for computing the rank. More sequential comparisons and some parallel ones are reported in [8]. 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 3 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 [10] 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 groups by Frank Heckenbach. The latter is a variant of the classical elimination method over arbitrary precision integers for Smith form [15], 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 [20] is computer algebra system for Macs and Windows. Its Smith form routine is an implementation of Kannan & Bachem’s [15] as is Heckenbach’s but with a different pivot strategy.

“Hom-Elim-GMP” and “Valence” ran on the 250 MHz Ultra-IL processor with 1024 Mb, but Fermat is only available on Mac and Windows. We therefore report on experiments with Fermat on a 400 MHz Intel 860 processor with only 320 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 - 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 2 we can see some of these effects.

6. 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.

7. REFERENCES


Table 3: Fermat vs. Hom-Elim-GMP vs. SFV


