An engineered algorithm for the Smith form of an integer matrix

B. David Saunders and Zhendong Wan
Department of Computer and Information Sciences
University of Delaware

A variety of algorithms for computing Smith normal forms of integer matrices are known. Their worst case asymptotic complexities have steadily improved over the past four decades. However, in practice, an asymptotically inferior algorithm often outperforms an asymptotically better one.

We offer an "engineered" algorithm for Smith forms of integer matrices, which is designed to combine the best aspects of previous algorithms. The fundamental base of our method is the Smith form algorithm of Eberly, Giesbrecht, and Villard. Our algorithm shares its worst case complexity. We provide several improvements such as a "bonus" idea which allows two adjacent invariant factors to be found at the price one, and a more practically efficient perturbation method. We "engineer in" judicious use of other algorithms in situations where they are more efficient. We discover the importance of seqarating rough and smooth parts of invariant factors and find an adaptive algorithm for computing each part. A number of experimental measurements suggest substantial benefit from our engineered Smith form algorithm.

Categories and Subject Descriptors: G.4 [**Mathematical Software**]: Algorithm design and analysis

General Terms: Algorithm, Experimentation

Additional Key Words and Phrases: Integer matrix, Smith normal form, invariant factor

1. INTRODUCTION

In 1861, H. M. S. Smith [Smith 1861] proved that any integer matrix can be diagonalized using elementary row and column operations. That is, for an $n \times n$ integer matrix A, there exist integer matrices U and V, and a unique diagonal matrix $S = diag(s_1, s_2, \ldots, s_n)$ with non-negative diagonal entries s_i, such that $A = USV$, $\det(U) = \pm 1$, $\det(V) = \pm 1$, and $s_i | s_{i+1}$, for $i = 1, \ldots, n-1$. The matrix S is called the Smith normal form or simply Smith form of A and its diagonal entries are called the invariant factors of A. It is a canonical form for matrix equivalence.

Over the past forty years, steady progress has been made in algorithms for computing the Smith forms. To the best of our knowledge, the asymptotically fastest algorithm for Smith forms given to date is in [Kaltofen and Villard 2003]. Using fast matrix multiplication and fast entry arithmetic, it requires a remarkably small $\mathrm{O}^{\sim}(n^{2.697263} \log_2(\|A\|))$ bit operations.

Supported by NSF grants 0098284 and 0112807

However, the most successful algorithm in current practice seems to be elimination mod the determinant as described by Iliopoulos/Hafner&McCurley[Iliopoulos 1989; Hafner and McCurley 1991]. The improvement of starting by computing the last invariant factor is studied in [Pan 1988; Abbott et al. 1999; Eberly et al. 2000; Storjohann 2005]. The last invariant factor can be computed effectively by using a fast rational solver [Moenck and Carter 1979; Dixon 1982]. Though this doesn't improve the worst case asymptotic complexity, it is important in cases where the Smith form is highly structured and the largest invariant factor is considerably smaller than the determinant. The method involves repeated alternations of unimodular row operations and unimodular column operations. In practice, typical matrices require a few repetitions with one row step and one column step completing the job. In such cases the cost behaves more like $\mathrm{O}^{\sim}(n^3(l + \log_2(\|A\|)))$. The banded reduction of a triangular matrix of Storjohann[Storjohann 1996] may be used. It has better complexity for the triangular case, but again for typical matrices most of the cost is experienced in an initial triangulation so that the speedup from this will be relatively small. In any case, so far we have not yet incorporated this improvement in our implementation. It would improve the practical run time of our algorithm, but not affect the structure of our adaptive algorithm.

Another approach, the valence algorithm of Dumas, et al[Dumas et al. 2000] involves computation of the minimal polynomial. It is effective when that polynomial is of low degree. This method first determines the primes which may occur in the invariant factors then uses local Smith form eliminations (computation done mod p^e for some prime p and sufficiently large exponent e. The local computations cost $\mathrm{O}^{\sim}(n^3 \log(p^e))$, and have small constants in the complexity and very effective implementation, especially when p^e is of machine word size.

We develop an adaptive algorithm whose purpose is to exploit the ideas of these various algorithms by using them in cases where they apply best while assuring that our worst case complexity will be that of the EGV algorithm. Our main contribution may be to focus on separating the handling of small primes and large primes that occur in the invariant factors. We also improve the preconditioners for the rational solver steps and particularly important in practice is the "bonus" which gets two invariant factors for essentially the cost of one.

Both algorithm 2.1 and algorithm 2.4 are probabilistic algorithms and require a larger number of iterations to correctly compute the invariant factor regarding small primes than they do regarding large primes. For example, for a non-singular matrix, 20 iterations of exactly solving non-singular linear systems are required in order to compute the power of 2 occurring in the largest invariant factor with probability of $1 - \frac{1}{1000,000}$, while only 4 iterations are expected to correctly compute the largest invariant with the same probability regarding all primes larger than 100. We'll do better to handle the contribution of small primes and large primes in separate ways. To facilitate the discussion of this, we borrow from number theory the terminology of smooth and rough numbers.

Definition 1.1. Given a positive integer k, a number is *k-smooth* if it is a product of primes smaller than k and is *k-rough* if it is a product of primes no less than k.

Clearly, for any positive k, every integer i can be written uniquely as a product of a k-smooth number and a k-rough number.

In this paper, we use the following notations. we say $f(n) = \mathrm{O}^{\sim}(g(n))$, if $f(n) = g(n)^{1+\mathrm{o}(1)}$. For an integer matrix A, we use $\log_2(\|A\|)$ to denote the bit length of the entry of A with the largest absolute value. We use ω to denote the exponent for matrix multiplication. Using standard matrix multiplication, $\omega = 3$. Using the algorithm of Coppersmith and Winograd [Coppersmith and Winograd 1987], $\omega = 2.376$. $M(b)$ s used to denote the number of bit operations required to multiply two integers of bit length b. Using FFT -see e.g. [von zur Gathen and Gerhard 1999, Section:8.3], $M(b) = \mathrm{O}(b \log b \log\log b) = \mathrm{O}^{\sim}(b)$, and we assume it in this paper.

In the next section we thoroughly examine the algorithm of [Eberly et al. 2000], presenting our additions and improvements. In particular, the consequences of focus on computing k-rough parts are spelled out. Then section 3 is a presentation of the engineered Smith form algorithm, and section 4 presents experimental results on examples from practice as well as on a few families of constructed test data. In the conclusion we summarize and remark on the open question of a good engineered Smith form algorithm for large sparse matrices.

2. EBERLY-GIESBRECHT-VILLARD ALGORITHM WITH IMPROVEMENTS

In 2000, W. Eberly, M. Giesbrecht and G. Villard [Eberly et al. 2000] discovered a method to compute the Smith form of an $n \times n$ integer matrix A in $\mathrm{O}^{\sim}(n^{3.5} \log \|A\|)$ or $\mathrm{O}^{\sim}(n^{2+\omega/2} \log_2(\|A\|))$ bit operations when fast matrix multiplication is available. The largest invariant factor of a non-singular integer matrix can be computed by solving a few linear systems. Then invariant factors of any index can be computed by means of random perturbation. Finally, all distinct invariant factors can be found by using a binary search. We start by discussing the method computing the largest invariant factor with our improvement. Then we will present a new perturbation method for the lower indexed invariant factors.

2.1 Largest invariant factor with a "bonus" idea

This algorithm is a variation of the algorithm [Eberly et al. 2000, LargestInvariant-Factor]. Our contribution is the "bonus" idea which can be used to find the second largest invariant factor with little extra work.

ALGORITHM 2.1. *LargestInvariantFactor*
Input:

—*A, a non-singular $n \times n$ integer matrix.*

—*k, a rough/smooth crossover*

—*e, number of iterations, influences probability bound. $e = 2$ works well in practice.*

Output:

—*The k-rough part of s_n and s_{n-1} (probabilistic, see theorem 2.3).*

Procedure:

(*1*) *Set* $M := \min\big(6 + 2n(\log_2 n + \log_2(\|A\|)), k\big); \mathcal{L} := \{0, \ldots, M-1\}$.

(*2*) *Set* $s_n^{(0)} := 1$; $s_{n-1}^{(0)} := 0$.

(*3*) *For i from* 1 *to e do*

a. *Choose random columns,* $\vec{b}^{(2i-1)}$, *and* $\vec{b}^{(2i)} \in \mathcal{L}^{n\times 1}$.
b. *Solve* $A\vec{x}^{(2i-1)} = \vec{b}^{(2i-1)}$ *and* $A\vec{x}^{(2i)} = \vec{b}^{(2i)}$ *over the rational field.*
c. *Compute*

$$t^{(2i-1)} := \operatorname{lcm}(\operatorname{denom}(\vec{x}_1^{(2i-1)}), \ldots, \operatorname{denom}(\vec{x}_n^{(2i-1)}))$$

and

$$t^{(2i)} := \operatorname{lcm}\big(\operatorname{denom}(\vec{x}_1^{(2i)}), \ldots, \operatorname{denom}(\vec{x}_n^{(2i)})\big).$$

d. *Compute* $u^{(i)} = \frac{\gcd(t^{(2i-1)}, t^{(2i)})}{s_2(\left[t^{(2i-1)}\vec{x}^{(2i-1)} \;\; t^{(2i)}\vec{x}^{(2i)}\right])}$.
e. *Compute* $s_n^{(i)} := \operatorname{lcm}(s_n^{(i-1)}, t^{(2i-1)}, t^{(2i)})$.
f. *Compute* $s_{n-1}^{(i)} = \gcd(s_{n-1}^{(i-1)}, u^{(i)})$.

(4) Return the k-rough part of $s_n^{(e)}$ *and the k-rough part of* $s_{n-1}^{(e)}$.

Remark 2.2. In addition to the largest invariant factor, algorithm 2.1 computes the k-rough part of the second largest one with extra $\mathrm{O}^{\sim}(n^2 \log_2(\|A\|))$ bit operations. This is of practical importance, since large prime factors often occur only in the largest invariant factor. For example, the random matrices have trivial invariant factors, i.e. all invariant factors are 1 except the largest one. Also the (second) largest invariant factor can be used to improve Iliopoulos' algorithm and Storjohann's, by using it in place of the determinant.

THEOREM 2.3. *Given an* $n\times n$ *non-singular integer matrix* A *and a rough/smooth crossover* k*, algorithm 2.1 computes the* k*-rough part of the largest invariant factor of* A *with probability at least* $1 - \big(1 + 4^e/(2e-1)\big)k^{1-2e}$ *the* k*-rough part of the second largest one of* A *with probability at least* $1 - \big(2^e + 4^e/(e-1)\big)k^{1-e}$. *The algorithm has bit complexity of* $\mathrm{O}^{\sim}(n^3 \log_2(\|A\|))$.

PROOF. In [Abbott et al. 1999], it is shown that for all i, $t^{(i)}$ is always a factor of s_n and for any prime $p \mid s_n$,

$$\operatorname{Prob}(p \mid \frac{s_n}{t^{(i)}}) \le \frac{1}{M}\lceil\frac{M}{p}\rceil$$

.

$2e$ independent columns, $\vec{b}$, are used in algorithm 2.1, so the error probability of computing the k-rough part of the largest invariant factor is at most:

$$\begin{aligned} & \textstyle\sum_{p\mid s_n, p>k}\big(\frac{1}{M}\lceil\frac{M}{p}\rceil\big)^{2e} \\ \le\; & \textstyle\sum_{p\mid s_n, k<p<M}\big(2/p\big)^{2e} + \sum_{p\mid s_n, p\ge M}\big(1/M\big)^{2e} \\ \le\; & \big(1 + 4^e/(2e-1)\big)k^{1-2e} \end{aligned}$$

Next we show that the computed $s_{n-1}^{(2)}$ is, with high probability, the k-rough part of the second largest invariant factor.

It is known that $s_n(A)A^{-1}$ is an integer matrix, and its Smith normal form is

$$\operatorname{diag}\big(s_n(A)/s_n(A), s_n(A)/s_{n-1}(A), \cdots, s_n(A)/s_1(A)\big).$$

Thus $S_{n-1}(A) = s_n(A)/s_2\big(s_n(A)A^{-1}\big)$.

Let C denote the integer matrix $s_n(A)A^{-1}$. For random columns $b^{(1)}$ and $b^{(2)}$ chosen from $\mathcal{L}^{n\times 1}$, by a general statement of theorem [Newman 1972, Theorem II.14], $s_2(C\left[b^{(1)}\ b^{(2)}\right])$ is always a factor of $s_2(C)$. From lemma 2.15 and a probability analysis similar to the proof of theorem 2.11, we know that for any prime p, [took out nmid here]

$$\mathrm{Prob}\big(p\frac{s_2\big(C\left[b^{(1)}\ b^{(2)}\right]\big)}{s_2(C)}\big) \geq (1-\frac{1}{M}\lceil\frac{M}{p}\rceil)(1-(\frac{1}{M}\lceil\frac{M}{p}\rceil)^2).$$

By the choice of $x^{(2i-1)}$ and $x^{(2i)}$, We know

$$s_n(A)A^{-1}\left[\vec{b}^{(2i-1)}\ \vec{b}^{(2i)}\right] = \left[s_n(A)\vec{x}^{(2i-1)}\ s_n(A)\vec{x}^{(2i)}\right].$$

Thus,

$$\frac{s_n(A)}{s_2\big(\left[s_n(A)\vec{x}^{(2i-1)}\ s_n(A)\vec{x}^{(2i)}\right]\big)} = \frac{\gcd(t^{(2i-1)},t^{(2i)})}{s_2\big(\left[t^{(2i-1)}\vec{x}^{(2i-1)}\ t^{(2i)}\vec{x}^{(2i)}\right]\big)}.$$

Therefore, the error probability of computing the k-rough part of the second largest invariant factor with two iterations is at most:

$$\begin{aligned}
&\textstyle\sum_{p|s^{(2)}_{n-1},p>k}\big(\frac{1}{M}\lceil\frac{M}{p}\rceil+(\frac{1}{M}\lceil\frac{M}{p}\rceil)^2\big)^e\\
\leq\ &\textstyle\sum_{p|s^{(2)}_{n-1},p>k}\big(\frac{2}{M}\lceil\frac{M}{p}\rceil\big)^e\\
\leq\ &\textstyle\sum_{p|s^{(2)}_{n-1},k<p\leq M}\big(\frac{4}{p}\big)^e+\sum_{p|s^{(2)}_{n-1},p>M}\frac{2^e}{M^e}\\
\leq\ &\big(2^e+4^e/(e-1)\big)k^{1-e}
\end{aligned}$$

Each iteration requires $\mathrm{O}^{\sim}(n^3\log_2(\|A\|))$ bit operations. Thus the bit complexity for algorithm 2.1 is $\mathrm{O}^{\sim}(n^3\log_2(\|A\|))$. □

2.2 An efficient perturbation method

Algorithm [Eberly et al. 2000, Algorithm:OneInvariantFactor] can be used to obtain the invariant factor of any index by means of perturbation. For $n\times i$ random matrix U and $i\times n$ random matrix V, the largest invariant factor of $A+UV$ is likely to be the ith invariant factor of A. We propose a more efficient perturbation method. Let L' and R' be random matrices with each entry randomly and independently chosen from a set of integers. Then the largest invariant factor of

$$\left[I_i\ L'\right]A\begin{bmatrix}I_i\\R'\end{bmatrix}$$

is likely to be the ith one of A. Additionally, the "bonus" idea in algorithm 2.1 can be extended here to compute the preceding one.

ALGORITHM 2.4. *OneInvariantFactor*
Input:

—*A, an $n\times m$ integer matrix, w.l.g. $m\leq n$.*
—*i, an integer, $2\leq i\leq$ rank(A).*
—*$k>10$, a smooth/rough crossover.*
—*e, number of iterations (influences probability bound). $e=4$ works well in practice.*

Output:

—The k-rough part of s_i and s_{i-1} (probabilistic, see theorem 2.6).

Procedure:

(1) *Set* $s_i^{(0)} := 0$.

(2) *Set* $s_{i-1}^{(0)} := 0$.

(3) *Set* $M := 210\lceil \frac{i(6\log n + 3\log_2(\|A\|))}{210} \rceil$; $\mathcal{L} = [0, \ldots, M-1]$.

(4) *For* $j = 1$ *to* e *do*

a. *Choose random integer matrix* $L \in \mathcal{L}^{i\times(n-i)}$.

b. *Choose random integer matrix* $R \in \mathcal{L}^{(m-i)\times i}$.

c. *Compute* $A' = \begin{bmatrix} I_i & L \end{bmatrix} A \begin{bmatrix} I_i \\ R \end{bmatrix}$.

d. *Compute* $(t^{(j)}, u^{(j)})$, *the largest and the second largest invariant factors of* A', *by algorithm 2.1.*

e. *Compute* $s_i^{(j)} := \gcd(s_i^{(j-1)}, t^{(j)})$; $s_{i-1}^{(j)} := \gcd(s_{i-1}^{(j-1)}, u^{(j)})$.

(5) *Return the k-rough part of* $s_i^{(e)}$, *and the k-rough part of* $s_{i-1}^{(e)}$.

Remark 2.5. The algorithm is similar to the algorithm in [Eberly et al. 2000, section 3]. But one random set is used here (rather than two), our new method is more practically efficient, and we are going to provide a tighter probabilistic lower bound.

THEOREM 2.6. *For a given* $n \times m$ *integer matrix* A, *an integer* i *with* $2 \le i \le$ *rank*(A), *a rough/smooth crossover* k, *and an iteration count* e, *algorithm 2.4 computes the k-rough part of the i-th invariant factor of* A *with probability at least* $1 - \big(2.24^{2e} + 4.48^{2e}/(2e-1)\big)k^{1-2e}$ *and the k-rough part of the* $(i-1)$*th one with probability at least* $1 - \big(4.48^{e} + 8.96^{e}/(e-1)\big)k^{1-e}$. *The algorithm above has bit complexity of* $\mathrm{O}^{\sim}(in^2\log_2(\|A\|))$.

The proof of theorem 2.6 is section 2.2.2, after some needed lemmas and theorems.

2.2.1 *Nearly uniformly distributed random variables.* In this subsection, we focus on non-uniformly but nearly uniformly distributed random variables over a finite field $\mathbb{F}$. If a distribution satisfies the bound condition,

$$\alpha \le \mathrm{Prob}(x = d) \le \beta, \forall d \in \mathbb{F},$$

we call it an $\{\alpha, \beta\}$-*distribution*. If $\alpha = \beta$, then x is a uniformly distributed random variable. The next lemma describes an example.

LEMMA 2.7. *If an integer* x *that is uniformly chosen from* $[0, \ldots, M-1]$ *is naturally mapped to a finite field* $\mathbb{Z}_p$, *then*

$$\alpha = \frac{1}{M}\lfloor\frac{M}{p}\rfloor \le \mathrm{Prob}(x = d \mod p) \le \frac{1}{M}\lceil\frac{M}{p}\rceil = \beta, \text{ for any } d \in \mathbb{Z}_p.$$

Thus $x \mod p$ *is an* $\{\alpha, \beta\}$-*distributed random variable over* $\mathbb{Z}_p$.

PROOF. Given an integer d and prime p, there are at most $\lceil \frac{M}{p} \rceil$ integers x such that,

$$0 \leq x \leq M-1, \text{ and } x = d \pmod{p}.$$

So $\text{Prob}(x = d \mod p) \leq \frac{1}{M}\lceil \frac{M}{p} \rceil$. By similar reasoning, $\text{Prob}(x = d \mod p) \geq \frac{1}{M}\lfloor \frac{M}{p} \rfloor$. □

The next lemma characterizes the distribution of dot product of a fixed non-zero vector and a random vector.

LEMMA 2.8. *Given a non-zero vector $\vec{t} \in \mathbb{F}^n$ and an element $d \in \mathbb{F}$, if $\vec{x} = (x_1, \ldots, x_n)$ is a random vector in $\mathbb{F}^n$ with each entry independently $\{\alpha, \beta\}$-distributed, then*

$$\alpha \leq \text{Prob}(\vec{x} \mid \vec{t} \cdot \vec{x} = d) \leq \beta.$$

PROOF. The right hand inequality can be proved by using the variation of the Schwartz-Zippel lemma in [Giesbrecht 2001, lemma2.1]. It can be argued in the following straightforward way, which applies as well to the left hand inequality.

Since $\vec{t} = (t_1, \ldots, t_n)$ is non-zero, without loss of generality, we assume $t_1 \neq 0$.

$$\begin{aligned}
&\text{Prob}(\vec{x} \mid \vec{t} \cdot \vec{x} = d) \\
=\ &\textstyle\sum_{j \in \mathbb{F}} \text{Prob}((x_2, \ldots, x_n) \mid \sum_{i=2}^n t_i x_i = j) \text{Prob}(x_1 \mid x_1 = \frac{d-j}{t_1} | \sum_{i=2}^n t_i x_i = j) \\
=\ &\textstyle\sum_{j \in \mathbb{F}} \text{Prob}((x_2, \ldots, x_n) \mid \sum_{i=2}^n t_i x_i = j) \text{Prob}(x_1 \mid x_1 = \frac{d-j}{t_1}) \\
\leq\ &\textstyle\sum_{j \in \mathbb{F}} \beta \text{Prob}((x_2, \ldots, x_n) \mid \sum_{i=2}^n t_i x_i = j) = \beta.
\end{aligned}$$

By similar reasoning, we know $\alpha \leq \text{Prob}(\vec{x} \mid \vec{t} \cdot \vec{x} = d)$. □

Next we characterize the distribution of a random null space vector.

LEMMA 2.9. *Given a matrix $A \in \mathbb{F}^{n \times m}$ with rank r, if $\vec{x} = (x_1, \ldots, x_m)^T$ is a random column in $\mathbb{F}^{m \times 1}$ with each entry independently $\{\alpha, \beta\}$-distributed, then*

$$\alpha^r \leq \text{Prob}(\vec{x} \mid Ax = 0) \leq \beta^r.$$

PROOF. We know there exit L, U, and P, such that $A = LUP$, P is a permutation matrix, L is non-singular, and U has the following shape

$$\begin{bmatrix} I_r & U' \\ 0 & 0 \end{bmatrix}.$$

Thus,

$$\begin{aligned}
&\text{Prob}(\vec{x} \mid Ax = 0) \\
=\ &\text{Prob}(\vec{x} \mid LUPx = 0) \\
=\ &\text{Prob}(\vec{x} \mid LUPx = 0) \\
=\ &\text{Prob}(\vec{x} \mid Ux = 0) \\
=\ &\text{Prob}((x_1, \ldots, x_r)^T = -U'(x_{r+1}, \ldots, x_n)^T) \\
=\ &\textstyle\prod_{j=1}^r \text{Prob}(x_j = -U_j'\dot{(}x_{r+1}, \ldots, x_n)^T | x_k = -U_k'\dot{(}x_{r+1}, \ldots, x_n)^T, \forall k, k < j) \\
=\ &\textstyle\prod_{j=1}^r \text{Prob}(x_j = -U_j'\dot{(}x_{r+1}, \ldots, x_n)^T) \leq \beta^r
\end{aligned}$$

By similar reasoning, $\alpha^r \leq \text{Prob}(\vec{x} \mid Ax = 0)$. □

More generally, for a given $b \in \mathbb{F}^m$, if we assume the same hypotheses, then

$$\text{Prob}(\vec{x} \mid Ax = b) \leq \beta^r.$$

Furthermore, if $Ax = b$ has a solution. then

$$\text{Prob}(\vec{x} \mid Ax = b) \geq \alpha^r.$$

The next lemma characterizes the distribution of a random vector lying in a subspace.

LEMMA 2.10. *Given a $k-$dimensional subspace of $\mathbb{F}^n$, $S = \text{Span}(S_1, \ldots, S_k)$, where $S_1, \ldots, S_k$ is a basis of S. If $\vec{x} = (x_1, \ldots, x_n)^T$ is a random vector in $\mathbb{F}^n$ with each entry independently $\{\alpha, \beta\}$-distributed, then*

$$\alpha^{n-k} \leq \text{Prob}(\vec{x} \mid x \in S) \leq \beta^{n-k}.$$

PROOF. We extend $(S_1, \ldots, S_k)$ to be a basis of $\mathbb{F}^n$, $(S_1, \ldots, S_n)$. Let $B = (S_1, \ldots, S_n)$. Then

$$\text{Prob}(\vec{x} \mid \vec{x} \in S) = \text{Prob}(\vec{x} \mid B^{-1}_{k+1,\ldots,n}\vec{x} = 0) \leq \beta^{n-k}.$$

By similarly reasoning, $\alpha^{n-k} \leq \text{Prob}(\vec{x} \mid \vec{x} \in S)$. □

The next theorem characterizes the distribution of a random non-singular matrix.

THEOREM 2.11. *Given an $n \times n$ matrix B over $\mathbb{F}$, if R is an $n \times n$ random matrix over $\mathbb{F}$ with each entry independently $\{0, \beta\}$-distributed, then*

$$\text{Prob}(\det(R - B) \neq 0) \geq \prod_{i=1}^{\infty}(1 - \beta^i).$$

PROOF. We will only prove the case where $B = 0$, since $B \neq 0$ follows from this case and lemma 2.8.

If we let $R = (R_1, \ldots, R_n)$, then $\det(A) \neq 0$ if and only if

$$a_i \notin \text{Span}(R_1, \ldots, R_{i-1}), 1 \leq i \leq n.$$

Thus,

$$\begin{aligned} & \text{Prob}(\det(R) \neq 0) \\ = \; & \textstyle\prod_{i=1}^{n} \text{Prob}(R_i \notin \text{Span}(R_1, \ldots, R_{i-1}) | R_1, \ldots, R_{i-1} \text{ are linearly independent.}) \; . \\ \geq \; & \textstyle\prod_{i=1}^{n}(1 - \beta^{n+1-i}) \geq \prod_{i=1}^{\infty}(1 - \beta^i). \end{aligned}$$

If $\beta < 1$, then $\prod_{i=1}^{\infty}(1 - \beta^i)$ converges and is positive. Moreover,

$$\prod_{i=1}^{\infty}(1 - \beta^i) = e^{\sum_{i=1}^{\infty} \ln(1-\beta^i)} \geq e^{\sum_{i=1}^{\infty}(-\beta^i)} \geq e^{\frac{-\beta}{1-\beta}} \geq 1 - \frac{\beta}{1-\beta}.$$

□

For the case where $B = 0$, there is a similar result in [Mulders and Storjohann 2004]. Note that in the uniform case over $\mathbb{F}_p$, we have $\beta = \frac{1}{p}$, thus

$$\text{Prob}(\det(R) \neq 0) \geq \prod_{i=1}^{\infty}(1 - (\frac{1}{p})^i).$$

Next we characterize the distribution of the perturbation.

THEOREM 2.12. *Given an $n \times n$ matrix T and an $m \times n$ matrix B over $\mathbb{F}$ with $rank(\begin{bmatrix} T \\ B \end{bmatrix}) = n$, if R is an $n \times n$ random matrix over $\mathbb{F}$ with each entry independently $\{0, \beta\}$-distributed, then*

$$\text{Prob}(R \mid \det(T + RB) \neq 0) \geq \prod_{i=1}^{\infty}(1 - \beta^i).$$

PROOF. If we let $r = rank(B)$, then there exists an $n \times n$ non-singular matrix Q, such that

$$BQ = \begin{bmatrix} B' & 0 \end{bmatrix},$$

where B' is of full column rank. Let us rewrite TQ as a block matrix, $\begin{bmatrix} T_1 & T_2 \end{bmatrix}$, where T_1 and T_2 are $n \times r$ and $n \times (n-r)$ matrices, respectively. Q is non-singular, so

$$n = \text{rank}(\begin{bmatrix} T \\ B \end{bmatrix}) = \text{rank}(\begin{bmatrix} TQ \\ BQ \end{bmatrix}) = \text{rank}(\begin{bmatrix} T_1 & T_2 \\ B' & 0 \end{bmatrix}).$$

Thus $\text{rank}(T_2) = n - r$. Therefore there exists a non-singular M, such that,

$$MTQ = \begin{bmatrix} T_{11} & 0 \\ T_{21} & T_{22} \end{bmatrix} \text{ and } T_{22} \text{ is non-singular,}$$

where T_{11}, T_{21} and T_{22} are $r \times r$, $r \times (n-r)$, and $(n-r) \times (n-r)$ matrices, respectively. Thus,

$$M(T + RB)Q = MTQ + MRBQ = \begin{bmatrix} T_{11} + M_1RB' & 0 \\ * & T_{22} \end{bmatrix}.$$

Therefore, it is evident that $\det(T + RB) \neq 0$ if and only $\det(T_{11} + M_1RB') \neq 0$, since M, Q, and T_{22} are all non-singular.

Note that M_1 has full row rank and B' has full column rank. There exist permutation matrices, P_1 and P_2, such that the $r \times r$ leading principle minors of M_1P_1 and P_2B' are non-singular. Let $R' = P_1^T R P_2^T$. Then $P_1RB' = M_1P_1R'P_2B'$. If we represent

$$M_1P_1 = \begin{bmatrix} M_{11} & M_{12} \end{bmatrix}, P_2B' = \begin{bmatrix} B_{11} & B_{12} \end{bmatrix}, R' = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix},$$

then

$$T_{11} + M_1RB' = M_{11}(R_{11} + Res(R_{12}, R_{21}, R_{22}, M_{11}, T_{11}, B_{11}))B_{11},$$

where

$$Res(R_{12}, R_{21}, R_{22}, M_{11}, T_{11}, B_{11}) = M_{11}^{-1}(T_{11} + M_{12}B_{21}B_{11} + M_{11}R_{22}B_{12} + M_{12}R_{22}B_{12})B_{11}^{-1}.$$

Since both M_{11} and B_{11} are non-singular, $T_{11} + M_1RB'$ is non-singular if and only if

$$\det\big(R_{11} + Res(R_{12}, R_{21}, R_{22}, T_{11}, M_{11}, B_{11})\big) \neq 0.$$

So $\det(T + RB) \neq 0$ if and only if

$$\det\big(R_{11} + Res(R_{12}, R_{21}, R_{22}, T_{11}, M_{11}, B_{11})\big) \neq 0.$$

Thus

$$\begin{aligned}&\text{Prob}(R \mid \det\big(R_{11} + Res(R_{12}, R_{21}, R_{22}, T_{11}, M_{11}, B_{11})\big) \neq 0)\\ &= \textstyle\sum_{C_1,C_2,C_3} \text{Prob}(R_{11} \mid \det\big(R_{11} + Res(C_1, C_2, C_3, T_{11}, M_{11}, B_{11})\big) \neq 0\\ &\quad |R_{12} = C_1, R_{21} = C_2, R_{22} = C_3)\text{Prob}(R_{12} = C_1, R_{21} = C_2, R_{22} = C_3)\\ &= \textstyle\sum_{C_1,C_2,C_3} \text{Prob}(R_{11} \mid \det\big(R_{11} + Res(C_1, C_2, C_3, T_{11}, M_{11}, B_{11})\big) \neq 0)\\ &\text{Prob}(R_{12} = C_1, R_{21} = C_2, R_{22} = C_3)\end{aligned}$$

Therefore, by theorem 2.11,

$$\begin{aligned}&\text{Prob}(R \mid \det\big(R_{11} + Res(R_{12}, R_{21}, R_{22}, T_{11}, M_{11}, B_{11})\big) \neq 0)\\ &\geq \textstyle\sum_{C_1,C_2,C_3} \cdot \prod_{i=1}^{\infty}(1-\beta^i)\text{Prob}(R_{12} = C_1, R_{21} = C_2, R_{22} = C_3)\ .\\ &\geq \textstyle\prod_{i=1}^{\infty}(1-\beta^i).\end{aligned}$$

So

$$\text{Prob}(R \mid \det(T + RB) \neq 0) \geq \prod_{i=1}^{\infty}(1-\beta^i).$$

□

Note that if T and B are $n \times n$ and $m \times n$ matrices over $\mathbb{F}$, respectively, and if there exists R, such that $T + RB$ is non-singular, then

$$\text{rank}(\begin{bmatrix}T\\B\end{bmatrix}) \geq \text{rank}(\begin{bmatrix}I & R\end{bmatrix}\begin{bmatrix}T\\B\end{bmatrix}) \geq \text{rank}(T + RB) = n,$$

therefore,

$$\text{rank}(\begin{bmatrix}T\\B\end{bmatrix}) = n.$$

COROLLARY 2.13. *Given a prime p, under the same assumption in the theorem above about T and B, if R is an $n \times n$ random integer matrix with each entry independently and uniformly chosen from $[0, \ldots, M-1]$, then*

$$\text{Prob}\big(\det(T + RB) \neq 0 \mod p\big) \geq \prod_{i=1}^{\infty}\big(1 - (\frac{1}{M}\lceil\frac{M}{p}\rceil)^i\big).$$

PROOF. This statement follows from theorem 2.12 and lemma 2.7. □

The lemma below explores the relationship between invariant factors of the perturbation and those of the original matrix.

LEMMA 2.14. *If L and A are $l \times n$, $n \times m$ integer matrices, respectively, then $s_i(LA)$ is always a multiple of $s_i(A)$, for all $1 \leq i \leq \min(l, m, n)$.*

PROOF. See [Newman 1972, Theorem II.14.] □

The lemma below explores the sufficient condition of $p \nmid \frac{s_i(LA)}{s_i(A)}$, where p is a prime.

LEMMA 2.15. *If A is an $n \times m$ integer matrix, p is a prime, and i is an integer, such that $s_i(A) \neq 0$, then there exists a full column ranked matrix M, such that for any $i \times n$ integer matrix L,*

$$p \nmid \frac{s_i(LA)}{s_i(A)}, \text{ if } \det(LM) \neq 0 \pmod p.$$

PROOF. Let e be the exponent of p in the largest invariant factor of A. Let us look at the Smith form of A mod p^{e+1}. There exist integer matrices U, V and D, such that,

$$A = UDV \pmod{p^{e+1}},\ \det(U) \neq 0 \pmod{p},\ \text{and } \det(V) \neq 0 \pmod{p},\ \text{and}$$

$$D \text{ is the Smith form of } A \text{ over } \mathbb{Z}_{p^{e+1}}.$$

We rewrite U, D as block matrices,

$$U = \begin{bmatrix} M & M' \end{bmatrix}, D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix},$$

where M is an $n \times i$ matrix, D_1 is an $i \times i$ matrix. Then

$$LA = \begin{bmatrix} LMD_1 & LM'D_2 \end{bmatrix} V.$$

Note that the entries of D_2 are multiples of those in D_1. So if $\det(LM) \neq 0 \mod p$, then D_1 is the Smith form of LA over $\mathbb{Z}_{p^{e+1}}$. i.e. $p \nmid \frac{s_i(LA)}{s_i(A)}$, and M has full column rank. □

2.2.2 *Proof of Theorem 2.6.* Based on lemmas and theorems above, we are able to prove theorem 2.6.

PROOF. By hypotheses, $s_i(A) \neq 0$. If L is a random $i \times (n-i)$ matrix with each entry uniformly and independently chosen from the integer set $[0, ..., M-1]$, then by lemma 2.15 and corollary 2.13,

$$\text{Prob}(L \mid p \nmid \frac{s_i(\begin{bmatrix} I_i & L \end{bmatrix} A)}{s_i(A)}) \geq \prod_{i=1}^{\infty}(1 - (\frac{1}{M}\lceil\frac{M}{p}\rceil)^i),\ \text{for any prime } p.$$

If the prime p is a factor of M, then $\frac{1}{M}\lceil\frac{M}{p}\rceil = \frac{1}{p}$, and approximate shows that the probability is at least $1 - \frac{1}{p-1}$ (see e.g. the proof of Theorem 2.11) and a careful one shows that the probability is at least $\sqrt{2}/5$ if $p = 2$ (see e.g. [Eberly 1997]). Note that $\frac{1}{M}\lceil\frac{M}{p}\rceil = \frac{1}{M}$, if $p \geq M \leq \frac{1}{p} + \frac{1}{M}$, otherwise. Therefore, if prime $11 < p \leq M$, then

$$\prod_{i=1}^{\infty}(1 - (\frac{1}{p} + \frac{1}{M})^i) \geq 1 - \frac{\frac{1}{p} + \frac{1}{M}}{1 - \frac{1}{p} - \frac{1}{M}} > 1 - \frac{\frac{1}{p} + \frac{1}{M}}{1 - \frac{1}{11} - \frac{1}{210}} > 1 - 1.12(\frac{1}{p} + \frac{1}{M})$$

and if $p \geq M$, then the probability is at least $1 - \frac{1}{M-1}$.

Thus, we know

$$\text{Prob}(L \mid p \nmid \frac{s_i(\begin{bmatrix} I_i & L \end{bmatrix} A)}{s_i(A)}) \geq \begin{cases} 1 - \frac{1}{M-1}, & \text{if } p \geq M \\ 1 - 1.12(\frac{1}{p} + \frac{1}{M}), & \text{if } 11 \leq p < M \\ \sqrt{2}/5, & \text{if } p = 2 \\ 1/2, & \text{if } p = 3 \\ 3/4, & \text{if } p = 5 \\ 5/6, & \text{if } p = 7 \end{cases}$$

Therefore, by reasoning similar to that in the proof of theorem 2.3, the error probability of computing the k-rough part of the ith invariant factor of A by algorithm

2.4 is at most:

$$\sum_{p|s_i^{(4)},k<p<M} \left(2 * 1.12(\frac{1}{p} + \frac{1}{M})\right)^{2e} \leq \left(2.24^{2e} + 4.48^{2e}/(2e-1)\right)k^{1-2e}.$$

By similar reasoning, the k-rough of the second largest invariant factor is computed by the bonus technique with error probability at most $\left(4.48^e + 8.96^e/(e-1)\right)k^{1-e}$.

In algorithm 2.4, each iteration costs

$$\mathrm{O}^{\sim}(in^2 + i^3(\log n + \log_2(\|A\|))).$$

So bit complexity of algorithm 2.4 is

$$\mathrm{O}^{\sim}(in^2 + i^3(\log n + \log_2(\|A\|))).$$

□

2.3 Binary search for invariant factors

Once there is a method to compute an arbitrary indexed invariant factor, a binary search can be used to find all distinct invariant factors.

THEOREM 2.16. *Given a $n \times n$ integer matrix and error bound ϵ, there exists a probabilistic algorithm which computes the Smith form of A with probability at least $1 - \epsilon$, and is expected to run in*

$$\mathrm{O}^{\sim}(n^{3.5}(\log_2(\|A\|))^{1.5}\log^2(1/\epsilon))$$

bit operations.

PROOF. See [Eberly et al. 2000, THEOREM 4.2]. □

3. AN ENGINEERED SMITH FORM ALGORITHM

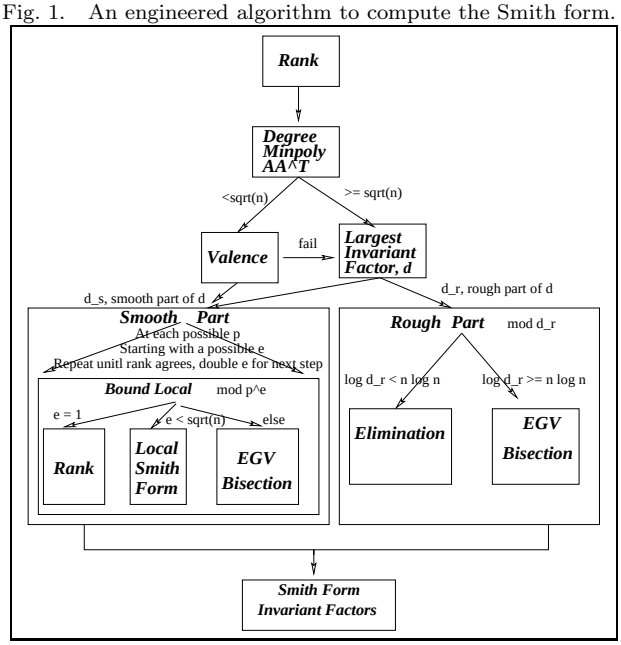
The Smith form algorithm in [Kaltofen and Villard 2003] has the best known asymptotic complexity. It has not been implemented. There is a good reason to expect it will be slow for feasible cases in the current hardware environment. The binary search algorithm in [Eberly et al. 2000] is also slow in the current hardware environment. On the other hand, earlier elimination algorithms such as in [Iliopoulos 1989; Hafner and McCurley 1991; Storjohann 1996] can be efficient in practice in many cases. Moreover, these algorithms can be improved by working mod the (second) largest invariant factor which can be computed either by algorithm 2.1 for a non-singular case or by algorithm 2.4 for a singular case. The computation of the largest invariant factor is practical. Based on our experiments, the computation of largest invariant factor with 4 rational solver calls takes only about 1/10 to 1/3 as long as the elimination step in typical non-singular examples. Also, we found that the valence algorithm in [Dumas et al. 2000] should be favored for certain matrices.

Based on these knowledge, an "engineered" algorithm for Smith form is proposed to deal with all cases. We called it *engineered* because its structure is based on both theoretical and experimental results. Different algorithms are suitable for different parts. In particular, we discovered the importance of separating the rough part and smooth part of invariant factors Therefore we divide the problem into smooth and

rough sub-problems. For each subproblem, we use an adaptive algorithm to solve it. The structure of the algorithm has been influenced by classes of matrices we have encountered in practice. We have designed it to recognize key patterns and go to the fastest practical algorithm for those cases, while avoiding cases where this could go badly awry.

The algorithm begins, for a matrix A, by computing r, its rank, and d, the degree of the minimal polynomial of A or AA^T. These can be done rapidly. The rank is needed in all cases, but d is computed just in order to decide between the valence approach and the other one which begins with the computation of the largest invariant factor. Next the smooth and rough parts of the valence or largest invariant factor are handled separately. The input to the smooth part includes a vector E of exponents for the first 25 primes. When coming from the valence, this is a zero-one vector with a zero indicating a prime known to be absent and a one indicating a prime which may occur. When coming from the largest invariant factor, the exponents are all positive and approximate the largest exponent of the prime occurring in the largest invariant factor. The organization of the adaptive Smith form algorithm is shown in figure 1.

Fig. 1. An engineered algorithm to compute the Smith form.



In practice, we choose k, the smooth/rough threshold to be 100. Note that there are exactly 25 primes which are smaller than 100. Below is an outline of our engineered Smith form algorithm:

ALGORITHM 3.1. *Engineered Smith form algorithm outline*
Input:

—A, *an* $n \times n$ *integer matrix.*

—ϵ, *an error probability requirement.*

Output:

—S, *vector of the invariant factors of* A. *Monte Carlo with probability at least* $1 - \epsilon$.

Procedure:

(*1*) *[rank] Compute* r = *rank of* A. *This can be done rapidly by computing mod a random prime from a suitable range. It is Monte Carlo with error probability at most* $\epsilon/2$.

(*2*) *[minpoly degree] Probablistically compute the degree of the minimal polynomial of* AA^T *(Computing the minimal polynomial mod a single random prime suffices for this step). If this degree is less than a threshold go to the valence step, otherwise proceed to the largest invariant factor step. We want a threshold less than* $\sqrt{n}$ *for asymptotic complexity reasons. In practice the valence method is advantageous primarily when the minimal polynomial degree is very low, so the threshold can be set lower than that.*

  (*a*) *[valence] Any prime that occurs in the invariant factors of* A *occurs in the valence (nonzero coefficient of lowest degree term) of the minimal polynomial of* A *or* AA^T . *It has been found in practice that some interesting matrices have low degree minimal polynomials and rapidly computed valence for* A *or* AA^T. *See [Dumas et al. 2000] for further discussion of the valence method and for examples.*
  Compute one of these valences, v. *If* v *is smooth, for* i *from 1 to 25, let* E_i *be 1 if the* i*-th prime divides* v, *and be* 0 *otherwise. Let* $t \rightarrow 1$, *and go to the smooth step. Otherwise the valence step fails, so proceed to largest invariant factor step.*

  (*b*) *[largest invariant factor] Compute the largest invariant factor* s_r *and bonus, the precedent one,* s_{r-1}, *using algorithm 2.1 or algorithm 2.4. The rough parts of the bonus and of the largest invariant factor agree with the true values with high probability. Thus the rough part of the bonus, often much smaller than the rough part of* s_r, *can be used in the rough part step below. Let* s *be the smooth part of the largest invariant factor for* i *from 1 to 25, let* E_i *be one greater than the exponent of the* i*-th prime in* s_r, *and let* t *be the remaining (rough) part of* s_r.

(*3*) *[smooth part] Let* $S_s \rightarrow diag(1, \ldots, 1)$, *this will become the smooth part of the Smith form. For* i *from 1 to 25, if* E_i *is positive, compute the local Smith form,* S_p *at the* i*-th prime, and set* $S_s \rightarrow S_s * S_p$. *The local Smith form at* p *is computed by starting with the estimated exponent* e. *If the number of non-zero invariant factors in* $A \pmod{p^e}$ *is less than* $\mathrm{rank}(A)$, *double* e, *repeat the computation until the number of non-zero invariant factors mod* p^e *is equal to* $\mathrm{rank}(A)$.

(4) *[rough part] Compute S_r the rough part of the Smith form by an elimination method if the modulus is small, or by a binary search if not. Correct the rough part of the largest invariant factor if necessary.*

(5) *[return] Compute and return $S = S_s * S_r$. In practice, S, S_s, and S_r may be represented as vectors (of the diagonal elements) or even as lists of the distinct entries with their multiplicities.*

The asymptotic complexity of the engineered algorithm above can be as good as the Smith form algorithm in [Eberly et al. 2000]. We have following theorem.

THEOREM 3.2. *There exist crossovers (thresholds) such that the engineered algorithm can probabilistically compute the Smith form of an $n \times n$ integer matrices in $\mathrm{O}^{\sim}(n^{3.5}(\log_2(\|A\|))^{1.5})$ bit operations.*

PROOF. In order to achieve $\mathrm{O}^{\sim}(n^{3.5}(\log_2(\|A\|))^{1.5})$, we need to choose crossovers such that each part can be computed in that bit complexity. Thus we choose the crossover between the valence and the largest invariant factor to be $\mathrm{O}^{\sim}(n^{0.5})$. Also we can choose the crossovers between an elimination method and a binary search method for the local Smith form and the rough part of the Smith form such that each can be computed in $\mathrm{O}^{\sim}(n^{3.5}(\log_2(\|A\|))^{1.5})$ bit operations. □

The constants in these crossover values are important in practice. Performance of the engineered algorithm is strongly influenced by them. In our implementation, these constants are chosen to favor the examples we've seen thus far in practice.

In the next section, we give some experimental results showing the effectiveness of this algorithm. Our experimentation is based on our implementation of the adaptive algorithm in the LinBox library. The code for generating random matrices used in our experiments is in the LinBox library. Other examples from other people are available upon request.

4. EXPERIMENTS

In this section, we report experiments based on matrices of interest in the study of finite structures, Chandler and Xiang[Chandler et al. 2005]; in number theory, Krattenthaler[Krattenthaler 2005; Almkvist et al. 2003]; and in group theory, Havas et al[Havas et al. 1993]. Also our design is influenced by matrices introduced to us by Welker [Dumas et al. 2000]. The latter are extremely sparse and the more interesting examples are too large for this dense matrix algorithm. Nonetheless, they also suggest that most Smith forms arising in practice do not involve large primes or at most involve large primes in the last one or two invariant factors. A future project is to extend [Dumas et al. 2000] to an engineered algorithm for dealing with sparse matrices.

4.1 Example matrices from applications

In work on incidence properties of affine planes over finite fields [Chandler et al. 2005], a number of Smith forms come up. We describe their general nature next, and indicate the structure of their Smith forms. The Smith forms are described in compact form as a list of value, occurrence pairs. A pair (v, n) denotes n occurrences of invariant v on the Smith form diagonal. The distinct invariant factors occurring are listed in increasing order.

C1093 and C9841 are (0,1)-matrices of the indicated orders, each having about 1/3 of the entries nonzero. In fact they are circulant, but we don't exploit that property here. They have smith forms whose invariant factors are powers of 3 (except the last which have additional factors 364 and 3280 respectively. For example, the Smith form of C9841 is [(1 145), (3 1440), (3^2, 1572), (3^3, 1764), (3^4, 1764), (3^5, 1572), (3^6, 1440), (3^7, 143), ($2^4 * 3^7 * 5 * 41$, 1)], and C1093's is similar.

C4369x70161 is again a (0,1)-matrix, expressing subspace incidences. It is quite sparse, with about 17 nonzero entries per column. The interest is in the power of 2 in the invariants, and, except for the last, all are relatively small powers of 2, which makes for very fast computation because the hardware overflow can be used as free normalization mod 2^{32}. The Smith from is [(1, 2801), (2, 640), (2^2, 416), (2^3, 256), (2^4, 255), ($2^4 \cdot 17$, 1)].

C5797 is a (0,1)-matrix with about 21 nonzeroes per row. The invariant factors in its Smith form are primarily powers of 2, but we see also some structure at 5 and, as in other examples, extra primes occurring only in the largest invariant. [(1, 2226), (2, 430), (2^2, 801), (2^3, 410), (2^4, 1590), ($2^4 \cdot 5$, 170), ($2^5 \cdot 5$, 70), ($2^6 \cdot 5$, 99), ($2^6 \cdot 3 \cdot 5 \cdot 7$, 1)]. We remark that, for this matrix, the Smith form computation seems to be a fast way to get the determinant. Computation of the determinant via eliminations mod primes and Chinese remaindering required about twice as long as the engineered Smith form algorithm.

Note that the bonus (second invariant factor) effectively removes need for further computation with the extra factors in the last invariant beyond the initial step of computing it. This can be a huge saving, most especially when the remaining part is smooth as so often happens in our experience.

Some smaller examples given to us by George Havas, arising in group theory, also exhibit this property of involving essentially one prime. In these examples, in fact, precisely one prime arises. They are H224x73 and H170x10. The first has entries of up to 3 digits and H170x10 has entries of size up to 8 digits. Their smith forms are [(1, 55), (5, 18)] and [(1, 2), (11, 8)] respectively. However, for these small examples with simple forms, it turns out that direct elimination over the integers is faster than first computing the largest invariant factor.

The Krattenthaler's matrices K1 through K20 arise in some number theoretical computation to find formulas for π[Almkvist et al. 2003]. K$\langle n \rangle$ has dimension $16n^2+8n+1$ and all entries are mostly of modest size. However a few entries are large. For example, K20 has 163 large entries of lengths ranging from 275 up to 1245 digits. Here the need is to compute the determinant (and factor it, which is easy since the factors are smooth). It turns out that these matrices also have rich structure in their Smith forms, meaning that numerous invariant factors are nontrivial and the largest invariant factor is significantly smaller than the determinant. For this reason, the smooth part of of our Smith form algorithm is the fastest way we know to compute these determinants.

We also experiments on a few constructed examples below:

(1) The matrix is randomly equivalent to $\oplus_{i=1}^{\sim \log(n)} F_i$, where F_i is a $i \times i$ tridiagonal {0,1,-1} matrix whose determinant is fib(i). For example, the Smith form for n = 100 case: $[(1, 96), (2, 1), (30, 1), (17160, 1), (7368166325520, 1)]$.

(2) The matrix is randomly equivalent to $\mathrm{diag}(1, 2, 3, \ldots, n)$. For example, the

Smith form for n = 100 case: $[(1,50),(2,17),(6,8),(12,5),(60,6),\ (420,2),$ $(840,1),\ (2520,2),\ (27720,2),\ (360360,1),\ (720720,1),\ (232792560,1),$ $(26771144400,1),\ (144403552893600,1),\ (3099044504245996706400,1),$ $(697203752297124771645338089353 12303556800,1)]$.

(3) The matrix is randomly equivalent to a diagonal matrix built solely from rough primes (bigger than 100). For example, the Smith form for n = 100 case: $[(1,82),(149,2),\ (20711,2),\ (2837407,2),\ (371700317,2),\ (7128096979109,2),$ $(805474958639317,2),\ (87796770491685553,2),\ (9394254442610354171,2),$ $(967608207588866479613,2)]$.

All experiments were run sequentially on a server with dual 3.2GHZ Xero processors, 6GB main memory. The elimination indicates the elimination step time by Iliopoulos' algorithm ignoring the determinant computation time. The Engineered indicated the run time computing the Smith form including computing the valences or the largest invariant factors. The Smith form algorithm [Eberly et al. 2000] turns out to be very slow in practice, except in the case where a matrix has all invariant factors equal.

Table I: Run time of our engineered algorithm over practical examples

matrix/runtime	C1093	$K\langle 10\rangle$	C5797	C9841	C4369x70161
engineered	47.03	7600.83	6445.45	42639.8	67558
Elimination	153.41	13224.2	20807.4	NEM	NEM

NEM stands for "not enough memory".

Table II: Runtime comparison over example class fib

Order	100	200	400	800	1600	3200
Engineered	0.42	2.23	14.56	179.65	1941.36	27266.6
Elimination	0.28	1.14	16.2	155.61	2175.38	33684.2

Table III: Runtime comparison over example class random

Order	100	200	400	800	1600	3200
Engineered	0.93	5.31	52.9	643.99	7497.65	133487
Elimination	1.08	8.28	85.82	1278.73	12881.5	185382

Table IV: Runtime comparison over example class random-rough

Order	100	200	400	800	1600	3200
Engineered	0.71	8.02	39.61	319.13	3225.96	48925.6
Elimination	0.57	4.26	44.1	470.66	5247.31	40511.9

We see in table 1 a speedup by a factor of 2 or 3 in the run times for matrices from practice as well as two cases in which our engineered algorithm succeeded while the elimination exausted available memory. The families of randomly constructed examples were designed to stress test the engineered algorithm. Nonetheless it performs well - tables 2, 3, and 4. For matrix order of 400 or greater it outperforms the direct elimination by anywhere from a few percentage points to a factor of 2. The one exception to this is the order 3200 random-rough example. In this case our hybrid uses about 20% more time, falling back to the elimination after having wasted time looking for other opportunities.

The source code for all these timings is available in LinBox at `http://linalg.org`. The algorithms compared are implemented in Linbox proper. The code for generating the random classes of matrices is in the examples directory. The matrices of Table 1 are available from the authors and from the collection of Jean-Guillaume Dumas at `http://www-lmc.imag.fr/lmc-mosaic/Jean-Guillaume.Dumas/simc.html`.

5. CONCLUSION

We have demonstrated that a practically useful engineered Smith form algorithm is feasible. Though it is designed particularly for large size problems, the overhead is slight in small problems. From a group of algorithms, each having advantages for some input matrices, this algorithm uses each in an advantageous way. This approach matches the best theoretical worst-case complexity of any of of the algorithms while also providing the best or near best run time on problems that have been encountered in practice. We believe this "engineering" approach can be profitably applied to other computer algebra problems.

A similar engineered algorithm optimized for Smith form on large *sparse* matrices would be very useful. However, there remain some open questions standing in the way of a good design for the sparse case, for instance memory efficient Smith form over Z_n, when n is a power of a small prime.

REFERENCES

Abbott, J., Brostein, M., and Mulders, T. 1999. Fast deterministic computation of determinants of dense matrices. In *Proc. ISSAC'99*. ACM Press, New York, 197 – 204.

Almkvist, G., Petersson, J., and Krattenthaler, C. 2003. Some new formulas for pi. *Experiment. Math. 12*, 441–456.

Chandler, D., Sin, P., and Xiang, Q. 2005. The invariant factors of the incidence matrices of points and subspaces in $pg(n, q)$ and $ag(n, q)$.

Coppersmith, D. and Winograd, S. 1987. Matrix multiplication via arithmetic progressions. In *Proc. 19th ACM STOC*. 1 – 6.

Dixon, J. D. 1982. Exact solution of linear equations using p-adic expansion. *Numer. Math.*, 137–141.

Dumas, J.-G., Saunders, B. D., and Villard, G. 2000. Smith form via the valence: Experience with matrices from homology. In *Proc. ISSAC'00*. ACM Press, 95 – 105.

Eberly, W. 1997. Processor-efficient parallel matrix inversion over abstract fields: Two extensions. In *Proc. 2nd PASCO*. 38–45.

Eberly, W., Giesbrecht, M., and Villard, G. 2000. On computing the determinant and Smith form of an integer matrix. In *Proc. 41st FOCS*. 675 – 687.

Giesbrecht, M. 2001. Fast computation of the Smith form of a sparse integer matrix. *Computational Complexity 10*, 41–69.

Hafner, J. L. and McCurley, K. S. 1991. Asymptotically fast triangularization of matrices over rings. *Siam J. Comput. 20*, 1068–1083.

Havas, G., Holt, D., and Rees, S. 1993. Recognizing badly presented Z-modules. *Linear algebra and its applications 192*, 137–163.

Iliopoulos, C. 1989. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups of the Hermite and Smith normal forms of an integer matrix. *SIAM J. Comput. Vol. 18, No.4*, 658 – 669.

Kaltofen, E. and Villard, G. 2003. On the complexity of computing determinants. Tech. Rep. 36, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon. October.

Krattenthaler, C. 2005. Advanced determinant calculus: a complement. *Linear Algebra Appl. 411*, 68–166.

Moenck, R. T. and Carter, J. H. 1979. Approximate algorithms to derive exact solutions to systems of linear equations. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*. Springer-Verlag, 65–73.

Mulders, T. and Storjohann, A. 2004. Certified dense linear system solving. *Jounal of symbolic computation 37(4)*.

Newman, M. 1972. *Integral Matrices*. Academic Press.

Pan, V. 1988. Computing the determinant and the characteristic polynomial of a matrix via solving linear systems of equations. *Information Processing Letters 28(2)*, 71–75.

Smith, H. M. S. 1861. On systems of indeterminate equations and congruences. *Philos. Trans.*, 293–326.

Storjohann, A. 1996. Near optimal algorithms for computing Smith normal forms of integer matrices. In *Proc. ISSAC'96*. ACM Press, 267 – 274.

Storjohann, A. 2005. The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity 21*, 609–650.

von zur Gathen, J. and Gerhard, J. 1999. *Modern Computer Algebra*. Cambridge University Press.