Proceedings of the 3rd ASCM, 1–6
Aug. 6–8, 1998. Lanzhou


# Matrix reduction algorithms for Euclidean rings

George Havas and Clemens Wagner
Centre for Discrete Mathematics and Computing
Department of Computer Science and Electrical Engineering
The University of Queensland, Queensland 4072, Australia
havas@csee.uq.edu.au
http://www.it.uq.edu.au/∼havas/
and
Rechenzentrum der Universität Karlsruhe
D-76128 Karlsruhe, Germany
clemens@exp-math.uni-essen.de
http://www.nic.de/clemens/

**Abstract.**
Reduction methods are the basis of algorithms for determining canonical forms of matrices over many computational domains. Experimental results have shown that various methods based on Gaussian elimination (which is a specialized kind of reduction algorithm) in Euclidean rings may lead to rapid growth of intermediate entries. On the other hand polynomial time algorithms do exist for such computations, but these algorithms are relatively complicated to describe and understand. Straightforward reduction methods provide the simplest descriptions of algorithms for this purpose. Such algorithms have a nice polynomial number of steps, but the steps may deal with operands with very large values. Here we show that there is a doubly exponential lower bound on the operands for a well-defined reduction algorithm when applied to Smith and Hermite normal form calculation for certain Euclidean rings. We present explicit matrices for which this variant produces doubly exponential entries. Thus, reduction algorithms have worst-case exponential space and time complexity for such applications. The analysis provides guidance as to how matrix algorithms for Euclidean rings which use reduction algorithms may be further developed for better performance, which is important since many practical algorithms for computing canonical forms are so based.

## 1. Introduction

A commutative ring $\mathcal{R}$ with identity 1 is Euclidean if there is a value function $\varphi : \mathcal{R}^* \to \mathbf{Z}_{\geq 0}$ (where $\mathcal{R}^* = \mathcal{R} \setminus \{0\}$ and $\mathbf{Z}_{\geq 0}$ is the set of nonnegative integers) such that the following properties hold for $a \in \mathcal{R}$ and $b \in \mathcal{R}^*$.

1. For $a \neq 0$, $\varphi(ab) \geq \varphi(a)$.
2. There exist $q, r \in \mathcal{R}$ with $a = qb + r$, such that either $r = 0$ or $\varphi(b) > \varphi(r)$.

Paradigm examples of Euclidean rings are $\mathbf{Z}$ (the ring of integers, with absolute value as value function) and $F[y]$ (the ring of univariate polynomials with coefficients in a field $F$, with degree as value function).

An element $a \in \mathcal{R}$ is a unit if it has an inverse $a^{-1} \in \mathcal{R}$ such that $aa^{-1} = a^{-1}a = 1$. The set $U(\mathcal{R})$ of all units of $\mathcal{R}$ is a multiplicative group. Elements $a, b \in \mathcal{R}$ are associates if there exists a unit $c \in \mathcal{R}$ such that $a = bc$ and we write $a \sim b$. Association is an equivalence relation with equivalence classes $[a] := \{b \in \mathcal{R} \mid a \sim b\}$. A subset $R \subseteq \mathcal{R}$ is a representative set for $\mathcal{R}$ if: $\{[a] \mid a \in R\} = \mathcal{R}/\sim$; and $\forall a, b \in R$, $a \not\sim b$.

For a Euclidean ring $\mathcal{R}$ a function $\rho : \mathcal{R} \times \mathcal{R}^* \to \mathcal{R}$ is called a residue class system if for all $a, a' \in \mathcal{R}$ and $b \in \mathcal{R}^*$

$$\begin{aligned} \rho(a,b) &\in \{a - qb \mid q \in \mathcal{R}\}, \\ \varphi(\rho(a,b)) &< \varphi(b), \qquad \text{and} \\ \rho(a,b) = \rho(a',b) &\iff \exists t \in \mathcal{R} : a = a' + tb. \end{aligned}$$

Let $M \subset \mathcal{R}$ with $M \neq \{0\}$ be a finite, nonempty subset of $\mathcal{R}$. The greatest common divisor of $M$ ($\gcd(M)$) is the equivalence class $[g]$ such that: $\forall a \in [g]$, $a \mid M$; and $\forall b \in \mathcal{R}$ with $b \mid M$, $b \mid [g]$. If we have a representative set $R$ for $\mathcal{R}$ and $d \in \gcd(M) \cap R$ then $d$ is uniquely determined. Further background material on Euclidean rings is given in [15, 5, 3].

Matrices $A$ and $B$ with entries in a Euclidean ring $\mathcal{R}$ are row equivalent if there exists a unimodular matrix $P$ such that $A = PB$. Matrix $P$ corresponds to a sequence of elementary row operations: multiplying a row by a unit of $\mathcal{R}$; adding any multiple by a ring element of one row to another; or interchanging two rows.

We use the following notation. For a $m \times n$ matrix $B$, say, we denote the entry in the $i$th row and $j$th column by $b_{i,j}$. We denote its $i$th row by $\mathbf{b}_{i*}$ and its $j$th column by $\mathbf{b}_{*j}$. When we wish to make the dimensions of the matrix clear we denote it by $B_{m\times n}$. Matrix $B$ may alternatively be written as

$$B = (\mathbf{b}_{*1}, \ldots, \mathbf{b}_{*n}) = \begin{pmatrix} \mathbf{b}_{1*} \\ \vdots \\ \mathbf{b}_{m*} \end{pmatrix}.$$

We denote by $||B||$ the maximum value of any entry in $B$.

For any matrix $B$ over a Euclidean ring $\mathcal{R}$ with representative system $R$ there exists a unique upper triangular matrix $H$ which is row equivalent to $B$ and which satisfies the following conditions. Let $r$ be the rank of $B$.

1. The first $r$ rows of $H$ are nonzero and the remaining rows are zero.
2. For $1 \leq i \leq r$ let $h_{i,j_i}$ be the first nonzero entry in row $i$. Then $j_1 < j_2 < \ldots < j_r$.
3. $h_{i,j_i} \in R \setminus \{0\}$, for $1 \leq i \leq r$.
4. For $1 \leq k < i \leq r$, $h_{k,j_i} = \rho(h_{k,j_i}, h_{i,j_i})$.

This matrix is called the row Hermite normal form (HNF) of the given matrix $B$ and has many important applications. There are many algorithms based on reduction methods for computing the HNF. Unfortunately such algorithms suffer from serious practical difficulties. Many of the problems which we address here are very similar to problems which arise in the related task of computing another canonical form of matrices over Euclidean rings, the Smith normal form (SNF).

Over the years different strategies have been proposed, primarily trying to avoid the major obstacle that occurs in such computations — explosive growth in size of intermediate entries. Methods for general Euclidean rings are considered in [20]. A comprehensive bibliography and a number of earlier methods for integer matrices are examined in [6], including references to various polynomial time algorithms. More recent methods are described in [7, 4, 19, 18, 10]. In [7, 10] the focus is on finding well-performing algorithms and heuristics for integer reduction methods. A worst case performance result for integers is given in [2]. Here we extend the main theorem of that paper to the most general possible environment.

## 2. A specific algorithm

Many descriptions of reduction methods for canonical form computation in Euclidean rings (sometimes for the integers) appear in the literature, including [15, 5, 3, 17]. In order to formally analyze the process we need to be quite explicit about the algorithm. We base our specific variant of an algorithm for HNF computation on pseudocode for integer Hermite normal form calculation due to Sims [17, p. 323], which provides a very general framework for reduction algorithms. Figure 1 gives a version of that algorithm for Euclidean rings, with line numbers for easy reference.

During execution of the algorithm there is **enormous** choice in selecting $k$ and $l$ in the while loop on lines 5 to 8. There is an exponential number of different execution sequences possible for a given input matrix depending on the choices made. Furthermore, the nature of the intermediate entries depends critically on the choices. Finding optimum choices is NP-hard in a well-defined sense, as reported in [7] for integer problems.

We define one step of the algorithm to comprise the work done in putting one column into final form. (This entails one execution of the code from lines 3 to 13.) Thus the first step creates an equivalent matrix to the input, with first column in HNF form.

```
    procedure ROWHNF (B, A) ;
    Euclidean B_{m×n} { input } ; A_{m×n} { output } ;
1   A := B ; i := 1 ; j := 1 ;
2   while i ≤ m AND j ≤ n do
      { check if rest of column j is zero }
3     if a_{k,j} = 0 for i ≤ k ≤ m then j := j + 1
4     else do
5       while ∃ k, l with i ≤ k ≠ l ≤ m
            such that a_{k,j} ≠ 0 and φ(a_{k,j}) ≤ φ(a_{l,j}) do
6         q := a_{l,j} div a_{k,j} ;
7         a_{l*} := a_{l*} − q × a_{k*}
8       endwhile
        { ∃! k, i ≤ k ≤ m such that a_{k,j} ≠ 0 }
9       interchange a_{i*} and a_{k*} ;
10      multiply a_{i*} by the unit to make a_{i,j} ∈ R ;
11      reduce the elements above a_{i,j} ;
12      i := i + 1 ; j := j + 1 ;
13    endif
14  endwhile
```

Fig. 1. *Pseudocode for Hermite normal calculation*

To complete the specification of our variant of the algorithm it suffices to indicate how $k$ and $l$ in the key loop, lines 5 to 8, are to be chosen. We do this in the next section, which includes our examples. (The implicit loop in line 11 is not needed for our analysis.)

## 3. Analysis

**Theorem 1** *Let $\mathcal{R}$ be either $\mathbf{Z}$ or $F[y]$. Given any integer $n > 0$ and a suitable $x \in \mathcal{R}^* \setminus U(\mathcal{R})$ there exists a $2(n+1)\times(n+1)$ matrix $A$ with entries in $\mathcal{R}$ and with $||A|| = \varphi(x+1)$ such that the maximal value intermediate entry in the working matrix has value at least $\varphi(x^{2^s}+1)$ during the $s$th step of our algorithm.*

**Proof.** For F[y], $x$ may be any nonzero, nonunit. For $\mathbf{Z}$, any $x > 1$ works with this example (and it can be readily extended to any nonzero, nonunit). We construct a matrix $A(n, x)$ for $n > 0$ as follows.

$$\text{Set } A(1,x) = \begin{pmatrix} 1 & -x \\ x & 1 \\ x & x \\ x & 0 \end{pmatrix}.$$

Then inductively define $A(i+1, x)$ by adding a 2-row, 1-column border:

$$A(i+1,x) = \begin{pmatrix} & 0 \\ & \vdots \\ A(i,x) & 0 \\ & x+1 \\ & 1 \\ x,0,\ldots,0, & x \\ x,0,\ldots,0, & 0 \end{pmatrix}.$$

Thus $A(n,x) =$

$$\left(\begin{array}{c|c|c|c|c|c|c|c}
1 & -x & 0 & 0 & 0 & 0 & 0 & \cdots \\
x & 1 & 0 & 0 & 0 & 0 & 0 & \\
x & x & x+1 & 0 & 0 & 0 & 0 & \\
x & 0 & 1 & 0 & 0 & 0 & 0 & \\
\hline
x & 0 & x & x+1 & 0 & 0 & 0 & \\
x & 0 & 0 & 1 & 0 & 0 & 0 & \\
\hline
x & 0 & 0 & x & x+1 & 0 & 0 & \\
x & 0 & 0 & 0 & 1 & 0 & 0 & \\
\hline
x & 0 & 0 & 0 & x & x+1 & 0 & \\
x & 0 & 0 & 0 & 0 & 1 & 0 & \\
\hline
x & 0 & 0 & 0 & 0 & x & x+1 & \\
x & 0 & 0 & 0 & 0 & 0 & 1 & \\
\hline
x & 0 & 0 & 0 & 0 & 0 & x & \\
x & 0 & 0 & 0 & 0 & 0 & 0 & \\
\hline
\vdots & & & & & & &
\end{array}\right)$$

The proof is by induction. Assume that after the $s$th step of the algorithm, column $s+1$ has the following form

$$\left(\begin{array}{c} * \\ \vdots \\ * \\ x^{2^s}+1 \\ x^{2^s}+x \\ x^{2^s} \\ \vdots \\ x^{2^s} \end{array}\right).$$

We make this true for $s = 1$ by defining part of our selection method for $k$ and $l$ in the key loop. Thus, if there exists $j$ such that $a_{n,j} \in U(\mathcal{R})$ choose $k = j$. (In our matrices there will be at most one such $j$ at any time.) Choose $l$ in an arbitrary fashion till the loop is completed. Thus, for the first column, the $a_{1,1}$ entry is used to set all other entries in that column to zero.

Now we can specify $k$ and $l$ for the first time in the key loop of the $(s+1)$st step of our algorithm. Choose $k = 2s+2$ (corresponding to the top entry $x^{2^s}$) and $l = 2s+1$ (corresponding to the entry $x^{2^s}+x$). This gives us a row, $\mathbf{a}_{(2s+1)*}$, with leading entry $x$. Now choose $k = 2s+1$ (corresponding to the entry $x$) and $l = 2s$ (corresponding to the entry $x^{2^s}+1$). This gives us a row, $\mathbf{a}_{(2s)*}$, with leading entry 1. Now choose $k = 2s$ (corresponding to the entry 1, as in the first step) for the remainder of this loop to eliminate (in any order) all remaining nonzero entries in this column. Note, in particular, what happens in the next column: in row $2s+2$ we have $x^{2^{s+1}}+1$; in row $2s+3$ we have $x^{2^{s+1}}+x$; and in all following rows we have $x^{2^{s+1}}$. This completes the induction and the proof. (The underlying theorem holds for arbitrary Euclidean rings.)

**Corollary 2** *Reduction methods for Hermite normal form calculation of matrices over the polynomial ring $F[y]$ will in the worst case generate intermediate polynomials with exponential degree.*

**Corollary 3** *Reduction methods have worst-case exponential space and time complexity for Hermite normal form calculation for integer matrices.*

**Proof.** The size of $A(n,2)$ is $\Omega(n^2)$. The reduction algorithm as described will generate doubly exponential entries in the working matrix (magnitude about $2^{2^s}$ in the $s$th step of the algorithm). These require exponential space to store and exponential time to compute, in terms of the size of the input. (This assumes standard binary representation for entries of the matrix. Clearly we could avoid the explosion in this case by working symbolically, as in the proof.)

**Corollary 4** *Reduction methods have worst-case exponential space and time complexity for Hermite normal form calculation for matrices over $Q[y]$, where $Q$ is the field of rationals.*

**Proof.** The size of $A(n, y+1)$ is $\Omega(n^2)$. The reduction algorithm as described will generate dense polynomials in the working matrix with exponential degree (about $2^s$ during the $s$th step of the algorithm). These require exponential space to store and exponential time to compute, in terms of the size of the input. (This assumes standard representation for entries of the matrix. $A(n, y)$ would not suffice to provide a bad example since we would expect a sensible sparse representation to be used for the polynomials. Again, we can avoid this explosion by working symbolically, as in the proof. However this is beyond the capabilities of current computer algebra systems, without intelligent assistance.)

**Corollary 5** *Reduction methods have worst-case exponential space and time complexity for Hermite normal form calculation for matrices over $GF(2^d)[y]$.*

**Proof.** Without loss of generality let $d = 1$. Consider $A(n, y+1)$. In the Proof for Theorem 1 we reduce the entry in row $2s$ with the entry in row $2s+1$. For this we have to compute $x^{2^s} + 1 - q * x$ where $q = x^{2^s-1}$ and with $x = y + 1$ we get $(y+1)^{2^s-1} = \sum_{i=0}^{2^s-1} \binom{2^s-1}{i} y^i = \sum_{i=0}^{2^s-1} y^i$ which is a dense polynomial with $2^s$ nonzero coefficents.

**Remarks.** We have written programs in various computer algebra languages which use the reduction algorithm that we specify in this paper. In all cases the space and time requirements of Corollaries 3, 4 and 5 are confirmed. Furthermore, this reduction algorithm generates exponentially sized entries in the corresponding transforming matrix.

## 4. Explanation

An explanation of the bad performance we see here comes from consideration of the extended gcd computation which comprises the first part of the calculation for the key loop. (Integer extended gcd calculation is studied in detail in [14, 8, 9, 16, 11].) Here we are computing the gcd of three values: $x^{2^n} + 1$, $x^{2^n} + x$ and $x^{2^n}$. We implicitly construct a solution vector plus a basis for the null-space. It is easy to see that our reduction algorithm gives as a solution to this problem

$$\begin{pmatrix} 1 & -x^{2^{n-1}} & x^{2^{n-1}} \\ -x & x^{2^n}+1 & -(x^{2^n}+1) \\ -x^{2^n} & x^{2^{n+1}-1} & -(x^{2^{n+1}-1}-1) \end{pmatrix} \begin{pmatrix} x^{2^n}+1 \\ x^{2^n}+x \\ x^{2^n} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

while an optimal solution is provided by

$$\begin{pmatrix} 1 & 0 & -1 \\ -x & 1 & x-1 \\ 0 & -x^{2^n-1} & x^{2^n-1}+1 \end{pmatrix} \begin{pmatrix} x^{2^n}+1 \\ x^{2^n}+x \\ x^{2^n} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Entries in each row of the poor solution are much larger than entries in the corresponding row of the good solution and are propagated through the working matrix.

It is worth contrasting these worst case results with the polynomial-time algorithms of Kannan and Bachem [12], Chou and Collins [1], and Havas, Majewski and Matthews [11] for the integers and of Kannan [13] for $Q[y]$. Those algorithms do not give careful attention to the extended gcd problem but instead achieve their polynomial complexity bounds by reducing matrix entries in additional processing steps.

## 5. Concluding Remarks

We have shown that the worst case behaviour of reduction algorithms for computing the Hermite normal form of a matrix with entries in a Euclidean ring has exponential space and time complexity. This result also applies to general row echelon form computation since we ignored the step of our algorithm (line 11) which normalizes the above-diagonal entries of the matrix. Likewise it applies to Smith normal form calculation: appropriate choices can be made in the SNF algorithm given in [5] so that our examples give the same explosion. (In fact for $n > 1$ the Hermite and Smith normal forms of $A(n, x)$ are the same, an $(n+1) \times (n+1)$ identity matrix above $(n+1)$ rows of zeros.)

The immediate cause of the entry explosion comes from an inefficient solution to the extended gcd problem being constructed for specific triples. The sorting gcd algorithm [8, 11, 20] produces close to optimal transforming

matrices for this three entry problem. This goes some way to explaining the better performance of matrix algorithms based on reduction methods which use sorting gcd principles. It also provides guidance as to how such algorithms may be further developed for better performance. This is important since many practical algorithms for computing canonical forms are so based, see [6, 7, 10, 20].

**Acknowledgements**

The first author was partially supported by the Australian Research Council.

## References

[1] T-W.J. Chou and G.E. Collins. Algorithms for the solution of systems of linear Diophantine equations. *SIAM J. Comput.* **11** (1982) 687–708.

[2] X.G. Fang and G. Havas. On the worst-case complexity of integer gaussian elimination. *ISSAC'97* (Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation), ACM Press (1997) 28–31.

[3] K.O. Geddes, S.R. Czapor and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, 1992.

[4] M. Giesbrecht. Fast computation of the Smith normal form of an integer matrix. *ISSAC'95* (Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation), ACM Press (1995) 110–118.

[5] B. Hartley and T.O. Hawkes. *Rings, Modules and Linear Algebra*. Chapman and Hall, 1976.

[6] G. Havas, D.F. Holt and S. Rees. Recognizing badly presented **Z**-modules. *Linear Algebra and its Applications* **192** (1993) 137–163.

[7] G. Havas and B.S. Majewski. Hermite normal form computation for integer matrices. *Congressus Numerantium* **105** (1994) 87–96.

[8] G. Havas and B.S. Majewski. Extended gcd calculation. *Congressus Numerantium* **111** (1995) 104–114.

[9] G. Havas and B.S. Majewski. A hard problem that is almost always easy. *Algorithms and Computation*, Lecture Notes Comput. Sci. **1004** (1995) 216–223.

[10] G. Havas and B.S. Majewski. Integer matrix diagonalization. *J. Symbolic Computation* **24** (1997) 399–408.

[11] G. Havas, B.S. Majewski and K.R. Matthews. Extended gcd and Hermite normal form algorithms via lattice basis reduction. *Experimental Mathematics* **7** (1998) 125–135.

[12] R. Kannan and A. Bachem. Polynomial algorithms for computing Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.* **8** (1979) 499–507.

[13] R. Kannan. Solving systems of linear equations over polynomials. *Theoretical Computer Science* **39** (1985) 69–88.

[14] B.S. Majewski and G. Havas. The complexity of greatest common divisor computations. *Algorithmic Number Theory*, Lecture Notes Comput. Sci. **877** (1994) 184–193.

[15] M. Newman. *Integral Matrices*. Academic Press, 1972.

[16] C. Rössner and J.-P. Seifert. The complexity of approximate optima for greatest common divisor computations. *Algorithmic Number Theory*, Lecture Notes Comput. Sci. **1122** (1996) 307–322.

[17] C.C. Sims. *Computation with finitely presented groups*. Cambridge University Press (1994).

[18] A. Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. *ISSAC'96* (Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation), ACM Press (1996) 267–274.

[19] A. Storjohann and G. Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. *ISSAC'96* (Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation), ACM Press (1996) 259–266.

[20] C. Wagner. Normalformberechnung von Matrizen über euklidischen Ringen. Dissertation, Fachbereich Mathematik und Informatik der Universität/GH Essen (1997).