BIT 17 (1977), 465–478

MULTIDIMENSIONAL GREATEST COMMON DIVISOR AND LEHMER ALGORITHMS

M. S. WATERMAN*

Abstract.

A class of multidimensional greatest common divisor algorithms is studied. Their connection with the Jacobi algorithm is established and used to obtain theoretical properties such as the existence of digit frequencies. A technique of D. H. Lehmer's for Euclid's algorithm is generalized for efficient computation of the multidimensional algorithms. For triples of integers, two algorithms of interest are studied empirically.

1. Introduction.

The efficient computation of greatest common divisors (gcds) has recently received attention, notably by Knuth [5]. One of the reasons for the interest in this subject is rational arithmetic where the integers tend to be large. The study of greatest common divisors and their efficient computation involves number theory, probability theory, and computer algorithms. This paper considers an algorithm stated by Knuth [5, p. 300] to compute the greatest common divisor of n integers. The connection of this algorithm with n-dimensional continued fractions will be utilized.

First, consider Euclid's algorithm for finding greatest common divisors. Given a pair of integers (m_1, m_2) where $m_1 < m_2$, the algorithm can be represented by

$$Q(m_1, m_2) = (m_2 \bmod m_1, m_1) = \left(m_2 - \left[\frac{m_2}{m_1}\right] m_1, m_1\right). \tag{1.1}$$

Of course,

$$\gcd(m_1, m_2) = \gcd Q(m_1, m_2)$$

so that the algorithm terminates when the first coordinate is 0. The second coordinate is then equal to $\gcd(m_1, m_2)$.

It will be useful to review the connection between Euclid's algorithm and continued fractions. Let

$$(m_1, m_2) \sim \frac{m_1}{m_2}$$

Received August 5, 1977.

* This work was partially supported by NSF grant #DCR75-07070.

associate each pair (m_1, m_2), $0 \leqq m_1 < m_2$, with a point in $[0, 1)$. Then

$$Q(m_1 m_2) = \left(m_2 - \left[\frac{m_2}{m_1}\right] m_1, m_1\right) \sim \frac{m_2}{m_1} - \left[\frac{m_2}{m_1}\right] = T\left(\frac{m_1}{m_2}\right)$$

where

$$T(x) = \frac{1}{x} - \left[\frac{1}{x}\right].$$

It is well known that T is the shift on the partial denominators of the continued fraction expansion of $x \in [0, 1)$. Thus, if $x = [a_1, a_2, \ldots]$, then $T(x) = [a_2, a_3, \ldots]$. It is important to notice that the integer $[m_2/m_1]$ in (1.1) for the first step of Euclid's algorithm is the first digit in the continued fraction expansion of m_1/m_2. Of course, this is true for the following digits as well.

In 1938, D. H. Lehmer [6] introduced a technique which is very useful for the efficient computation of gcd (m_1, m_2). Suppose m_1, m_2 are multiprecision integers and m_1', m_1'', m_2', m_2'' are single precision integers such that

$$\frac{m_1'}{m_2'} < \frac{m_1}{m_2} < \frac{m_1''}{m_2''}.$$

Then,

$$\frac{m_2''}{m_1''} < \frac{m_2}{m_1} < \frac{m_2'}{m_1'}.$$

If

$$\left[\frac{m_2''}{m_1''}\right] = \left[\frac{m_2'}{m_1'}\right] = a, \quad \text{then } \left[\frac{m_2}{m_1}\right] = a$$

and

$$\frac{m_2''}{m_1''} - a < \frac{m_2}{m_1} - a < \frac{m_2'}{m_1'} - a$$

or

$$T\left(\frac{m_1''}{m_2''}\right) < T\left(\frac{m_1}{m_2}\right) < T\left(\frac{m_1'}{m_2'}\right)$$

and the procedure can be repeated, doing arithmetic only with the single precision integers, until the a's disagree. Then a single multiprecision catch-up step can be performed. See Knuth [5, p. 305].

Next, Euclid's algorithm is generalized to an $n+1$ tuple of integers $\boldsymbol{m} = (m_1, m_2, \ldots, m_{n+1})$ with $n \geqq 1$. Suppose $m_i < m_{n+1}$ for $1 \leqq i < n+1$. Then define

$$\begin{aligned} Q(\boldsymbol{m}) &= (m_2 \bmod m_1, m_3 \bmod m_1, \ldots, m_{n+1} \bmod m_1, m_1) \\ &= \left(m_2 - \left[\frac{m_2}{m_1}\right] m_1, m_3 - \left[\frac{m_3}{m_1}\right] m_1, \ldots, m_{n+1} - \left[\frac{m_{n+1}}{m_1}\right] m_1, m_1\right). \end{aligned}$$

Again,

$$\gcd(\boldsymbol{m}) = \gcd Q(\boldsymbol{m}) .$$

Each m is associated with a point in $[0,1)^n$ by

$$m \sim \left(\frac{m_1}{m_{n+1}}, \frac{m_2}{m_{n+1}}, \ldots, \frac{m_n}{m_{n+1}}\right).$$

Now

$$(1.2)\quad Q(\boldsymbol{m}) \sim \left(\frac{m_2}{m_1} - \left[\frac{m_2}{m_1}\right], \ldots, \frac{m_{n+1}}{m_1} - \left[\frac{m_{n+1}}{m_1}\right]\right) = T\left(\frac{m_1}{m_{n+1}}, \ldots, \frac{m_n}{m_{n+1}}\right)$$

where

$$T(x_1, \ldots, x_n) = \left(\frac{x_2}{x_1} - \left[\frac{x_2}{x_1}\right], \ldots, \frac{x_n}{x_1} - \left[\frac{x_n}{x_1}\right], \frac{1}{x_1} - \left[\frac{1}{x_1}\right]\right).$$

The new transformation T is the shift on the partial denominators of the n-dimensional continued fraction of Jacobi and Perron [7]. Much is known about these continued fractions, including their convergence. The books by Leon Bernstein [1] and Fritz Schweiger [7] contain full accounts of the modern theory. However, not much attention seems to have been given to the problem of greatest common divisors and their connection with n-dimensional continued fractions.

For the purposes of computing greatest common divisors, however, Q is not the most efficient n-dimensional algorithm. Let the function O be a permutation defined on a vector by making the smallest component the first component of the new vector and leaving all other orders unchanged. Then define

$$(1.3)\qquad P(\boldsymbol{m}) = O(Q(\boldsymbol{m}))$$

and

$$S(\boldsymbol{x}) = O(T(\boldsymbol{x})) .$$

The transformation P allows the next step of the algorithm to divide by the smallest component of $Q(\boldsymbol{m})$, thereby reducing the largest component of $Q(P(\boldsymbol{m}))$ as compared to $Q(Q(\boldsymbol{m}))$.

In this paper the pairs of transformations (Q, T) and (P, S) will be studied. A theoretical basis will be stated for the algorithms, some problems associated with the computer algorithms will be solved, and some numerical studies will be presented. One result of special interest is the generalization of Lehmer's method to the n-dimensional situation. Lehmer's method has been generalized to the Gaussian integers [4] but this may be the first multidimensional generalization.

2. Properties of the Algorithms

A great deal is known about the shift, T, for the n-dimensional continued fraction. The emphasis here will be on the metric theory [10]. The shift, T, is ergodic and there exists a measure $\mu_T \sim \lambda$ (n-dimensional Lebesgue measure) such that T is measure preserving with respect to μ_T. Then, by the ergodic theorem [10],

$$\frac{1}{m}\sum_{i=0}^{m-1} \chi_{B(\boldsymbol{a})}(T^i(\boldsymbol{x})) \to \mu_T(B(\boldsymbol{a})) \tag{2.1}$$

for almost all $\boldsymbol{x} \in (0,1)^n$ where $\boldsymbol{a} = (a_1,\ldots,a_n)$ and $B(\boldsymbol{a}) = \{x \in (0,1)^n : \boldsymbol{a} = ([x_2/x_1], \ldots, [x_n/x_1], [1/x_1])\}$. Therefore (2.1) states that the long run average number of times, when $\boldsymbol{a}$ is obtained by iterations of T, is $\mu_T(B(\boldsymbol{a}))$ for almost all $\boldsymbol{x}$.

Recently, the metric theory of the transformation S was obtained by the author [10]. The continued fraction associated with S was shown to converge and the ergodic theory established. Consequently,

$$\frac{1}{m}\sum_{i=0}^{m-1} \chi_{B(\boldsymbol{a})}(S^i(\boldsymbol{x})) \to \mu_S(B(\boldsymbol{a})) \tag{2.2}$$

for almost all $x \in (0,1)^n$ where $\mu_S \sim \lambda$ is the invariant measure for S. It is interesting to note that for T the set of possible $\boldsymbol{a}$ satisfies

$$0 \leqq a_i \leqq a_n \quad \text{for all } i$$
$$1 \leqq a_n$$

while for S they satisfy

$$1 \leqq a_i \leqq a_n \quad \text{for all } i\,.$$

Of course the transformations Q and P defined in (1.2) and (1.3) imply that our interest is exactly in T and S defined on *rationals*, and the rationals are a set of Lebesgue measure zero. However, other work by the author [9] has considered this problem. Let $A_m = \{0/m, 1/m, \ldots, (m-1)/m\}^n$ and $C_k(\boldsymbol{a}) = \{\boldsymbol{x} \in (0,1)^n : a_{k+1}(\boldsymbol{x}) = \boldsymbol{a}\}$. Then

$$\lim_{k\to\infty}\lim_{m\to\infty} \frac{\#(C_k(\boldsymbol{a}) \cap A_m)}{m^n} = \mu(B(\boldsymbol{a})) \tag{2.3}$$

where μ is μ_T or μ_S.* This statement shows that rationals, on the average, have the same long run digit frequencies that almost all $\boldsymbol{x}$ have.

Thus, if approximate values of $\mu(B(\boldsymbol{a}))$ are obtained, an estimate of the normal size of a_i in $\boldsymbol{a} = (a_1,\ldots,a_n)$ can be made. If the a_i are very likely to fit in single precision words, the efficient computation can proceed without difficulty. For Euclid's algorithm, a_1 will be less than 1000 about 99.856 percent of the time [5, p. 305]. Estimates of digit frequencies for $n=2$ will be made in section 3.

* The symbol # denotes "the number of".

Next, the problem of generalizing Lehmer's method to n dimensions is considered. The solution was begun in a paper [2] on ergodic computation with Jacobi's algorithm for $n=2$. Also, the proof given there is incomplete. Define

$$\Psi(x_1,\ldots,x_n) = \left(\frac{x_2}{x_1},\ldots,\frac{x_n}{x_1},\frac{1}{x_1}\right).$$

Clearly

$$T(\boldsymbol{x}) = \Psi(\boldsymbol{x}) - a(\boldsymbol{x}) \tag{2.4}$$

where, as above, $a(\boldsymbol{x}) = ([x_2/x_1],\ldots,[x_n/x_1],[1/x_1])$. The relation (2.4) will be used to generalize Lehmer's method. First, a crucial lemma is proved. R. W. Johnson made the observation that Ψ maps hyperplanes into hyperplanes is not sufficient to establish Lehmer's method.

LEMMA 2.1. *The transformation Ψ takes any simplex in $(0,1)^n$ into a simplex.*

PROOF. Let $\alpha_0+\alpha_1x_1+\alpha_2x_2+\ldots+\alpha_nx_n=0$ be a hyperplane and $x_1 \neq 0$. Then $\alpha_0(1/x_1)+\alpha_1+\alpha_2(x_2/x_1)+\ldots+\alpha_n(x_n/x_1)=0$ which is a hyperplane in the image of Ψ. It is easy to check that Ψ is a continuous one-to-one mapping. Since a line can be obtained as an intersection of $n-1$ hyperplanes, Ψ maps lines into lines.

Suppose $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the simplex. Let $L(\boldsymbol{\alpha},\boldsymbol{\beta})$ denote the line segment from $\boldsymbol{\alpha}$ to $\boldsymbol{\beta}$. Now, $\Psi(\boldsymbol{x})$ and $\Psi(\boldsymbol{y})$ belong to $\Psi(L(\boldsymbol{x},\boldsymbol{y}))$, and $\Psi(L(\boldsymbol{x},\mathrm{y}))$ lies on a line. Since $L(\boldsymbol{x},\boldsymbol{y})$ is connected, $\Psi(L(\boldsymbol{x},\boldsymbol{y}))$ is connected and it follows that

$$L(\Psi(\boldsymbol{x}),\Psi(\boldsymbol{y})) \subset \Psi(L(\boldsymbol{x},\boldsymbol{y})).$$

But, if equality did not hold in the above set relation, the one-to-one property of Ψ would be violated. Therefore, Ψ maps line segments into line segments.

Any point in the simplex (in the domain) can be written as $\boldsymbol{x}=\sum \alpha_i\boldsymbol{x}_i$. By rewriting this relation as

$$\boldsymbol{x} = (\alpha_1+\alpha_2)\frac{\alpha_1\boldsymbol{x}_1+\alpha_2\boldsymbol{x}_2}{\alpha_1+\alpha_2}+\ldots,$$

it can be seen that

$$\Psi(\boldsymbol{x}) = \sum \beta_i\Psi(\boldsymbol{x}_i)$$

where $0 \leqq \beta_i$ and $\sum \beta_i = 1$. This completes the proof of the lemma.

Next Lehmer's method is generalized to the Jacobi algorithm.

THEOREM 2.1. *Let $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{n+1}$ belong to $(0,1)^n$ and let $\boldsymbol{x}$ belong to their closed convex hull. Assume $a(\boldsymbol{y}_1)=a(\boldsymbol{y}_2)=\ldots=a(\boldsymbol{y}_{n+1})$. Then $a(\boldsymbol{x})=a(\boldsymbol{y}_i)$ and $T(\boldsymbol{x})$ belongs to the closed convex hull of $T(\boldsymbol{y}_1),\ldots,T(\boldsymbol{y}_{n+1})$.*

PROOF. Lemma 2.1 states that Ψ maps any simplex in $(0,1)^n$ into a simplex. Thus $\Psi(\boldsymbol{x})$ belongs to the closed convex hull of $\Psi(\boldsymbol{y}_1),\ldots,\Psi(\boldsymbol{y}_{n+1})$. But if the integer parts of the coordinates of each $\Psi(y_i)$ are identical, then $a(x)=a(y_i)$ and $\Psi(x)-a(x)=T(x)$ must belong to the closed convex hull of $T(y_1)=\Psi(y_1)-a(y_1),\ldots,T(y_{n+1})=\Psi(y_{n+1})-a(y_{n+1})$.

The situation for the transformation S is slightly more complicated. Recall that

$$S(x) = O(T(x)) .$$

Let $\sigma(x)$ denote the permutation performed by O.

THEOREM 2.2. *Let $y_1; y_2,\ldots,y_{n+1}$ belong to $\{t\colon 0<t_1\leqq t_i\leqq 1$ for $1\leqq i\leqq n\}$ and let x belong to their closed convex hull. Assume $a(y_1)=a(y_2)=a(y_{n+1})$. Then $a(x)=a(y_i)$. If also $\sigma(y_1)=\sigma(y_2)=\ldots=\sigma(y_{n+1})$, then $\sigma(x)=\sigma(y_i)$ and $S(x)$ belongs to the closed convex hull of $S(y_1),\ldots,S(y_{n+1})$.*

PROOF. That $a(y_1)=\ldots=a(y_{n+1})$ implies $a(x)=a(y_i)$ follows from Theorem 2.1. Since convexity and interior points are preserved under permutation, $\sigma(y_1)=\ldots=\sigma(y_{n+1})$ implies that $\sigma(x)=\sigma(y_i)$. Finally, it is clear that $S(x)$ belongs to the permuted set.

From Theorems 2.1 and 2.2, it is clear that it is necessary to find $y_1,\ldots,y_{n+1}$ such that x belongs to the closed convex hull. Also, this must be done by some algorithm which can be programmed. This task is begun in the following theorem.

THEOREM 2.3. *Let x belong to $(0,1)^n$. Choose z in the same set so that $z_i<x_i$ for $1\leqq i\leqq n$ and define $(\varepsilon_i>0)$ $y_i=(z_1,\ldots,z_{i-1},z_i+\varepsilon_i,z_{i+1},\ldots,z_n)$ for $1\leqq i\leqq n$ and $y_{n+1}=z$. Then x belongs to the closed convex hull of $y_1,y_2,\ldots,y_{n+1}$ if and only if*

$$\sum_{i=1}^{n}\frac{(x_i-z_i)}{\varepsilon_i}\leqq 1 .$$

PROOF. Now x belongs to the closed convex hull of $y_1,\ldots,y_{n+1}$ if and only if

$$x = \sum_{i=1}^{n+1}\alpha_i y_i$$

where $\alpha_i\geqq 0$ and $\sum_{i=1}^{n+1}\alpha_i=1$.

But

$$\sum_{i=1}^{n+1}\alpha_i y_i = z+(\alpha_1\varepsilon_1\ldots,\alpha_n\varepsilon_n)$$

and this implies

$$(x_1-z_1,\ldots,x_n-z_n) = (\alpha_1\varepsilon_1,\ldots,\alpha_n\varepsilon_n)$$

which holds if and only if

$$\alpha_i = \frac{x_i-a_i}{\varepsilon_i},\qquad 1\leqq i\leqq n .$$

Then

$$\sum_{i=1}^{n} \frac{x_i - z_i}{\varepsilon_i} = 1 - \alpha_{n+1} \leqq 1 .$$

For the converse, let $\alpha_i = x_i - z_i/\varepsilon_i$, $1 \leqq i \leqq n$, and $\alpha_{n+1} = 1 - \sum_{i=1}^{n} \alpha_i \geqq 0$.

It must be pointed out that this does not yet solve the problem for computational purposes. The criteria of whether or not $\boldsymbol{x}$ belongs to the closed convex hull defined by $\boldsymbol{z}$ and $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$ is stated in terms of $\boldsymbol{x}$. The purpose of Lehmer's method is to deal with single precision $\boldsymbol{y}_i$ and avoid the multiprecision $\boldsymbol{x}$.

The problem, of course, is to deal with the coordinates of $\boldsymbol{x}$ when the coordinates are ratios of multiprecision integers. Let $0 < c \leqq d$ be multiprecision integers and, if d has β places base 10, let

$$e = [c/10^{\beta - w}], \quad f = [d/10^{\beta - w}] .$$

Thus e and f are the leading digits of c and d. For a lower bound on $x_i = c/d$ use $z_i = e/(f+1)$. Then to compute a bound on $x_i - z_i$, consider

$$0 < \frac{c}{d} - \frac{e}{f+1} < \frac{e+1}{f} - \frac{e}{f+1} = \frac{e+f+1}{f(f+1)}$$

$$= \frac{1 + e/f + 1/f}{f+1} \leqq \frac{2 + 1/f}{f+1} \leqq \frac{2 + 10^{-w}}{f+1} .$$

It is convenient to consider ε_i of the form $\varepsilon_i = \gamma/f+1$. It is sufficient, by Theorem 2.3, to have $x_i - z_i/\varepsilon_i \leqq 1/n$ so consider

$$\frac{n(2+10^{-\beta})}{f+1} \leqq \varepsilon_i = \frac{\gamma}{f+1} .$$

Therefore, $\gamma \geqq 2n + n10^{-w}$ is a sufficient number of places. Since γ should be an integer, use $\gamma = 2n + [n10^{-w}] + 1$.

For the numerical work reported on in section 3, $n = 2$ and $w = 9$ is often used. Then $\gamma \geqq 4 + 2 \cdot 10^{-9}$, so that for complete certainty $\gamma = 5$ should be used for finding y_1, y_2. However, the earlier approximations are usually conservative and $\gamma = 4$ can be used. To clarify these concepts, a simple example is given. Let $w = 4$ and

$$\boldsymbol{m} = (2718281828, 3141592654, 10000000000)$$

$$(\boldsymbol{x} = (.2718281828, .3141592654)) .$$

The point corresponding to $\boldsymbol{z}$ is

$$(2718, 3141, 10001)$$

and the points corresponding to $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ $(\gamma = 5)$ are

$$(2723, 3141, 10001)$$

and

$$(2718, 3146, 10001) .$$

Some additional results for the generalization of Lehmer's method are necessary. Above, a practical technique of generating a simplex containing $\boldsymbol{x}$ was presented. If the sequence of transformations and digits are denoted by

$$\boldsymbol{m} = \boldsymbol{m}^0$$

$$\boldsymbol{m}^1 = Q(\boldsymbol{m}),\ \boldsymbol{a}^1 = \boldsymbol{a} = \left(\left[\frac{m_2}{m_1}\right],\ldots,\left[\frac{m_{n+1}}{m_1}\right],\left[\frac{1}{m_1}\right]\right)$$

$$\boldsymbol{m}^2 = Q^2(\boldsymbol{m}),\ \boldsymbol{a}^2 = (a_1^2, a_2^2,\ldots,a_n^2)$$

$$\ldots$$

$$\boldsymbol{m}^k = Q^k(\boldsymbol{m}),\ \boldsymbol{a}^k = (a_1^k, a_2^k,\ldots,a_n^k)\,,$$

then Lehmer's method can assure that, say, all k of these digits $\boldsymbol{a}^1,\ldots,\boldsymbol{a}^k$ are correct. However, for the process to continue to step $k+1$, the vector $Q^k(\boldsymbol{m})$ is required.

To obtain $Q^k(\boldsymbol{m})$ from $\boldsymbol{a}^1,\ldots,\boldsymbol{a}^k$ and $\boldsymbol{m}$, notice that

$$m_1^0 = (1,0,\ldots,0)\cdot\boldsymbol{m}^0 = \boldsymbol{c}_1^0\cdot\boldsymbol{m}^0\,,$$

$$\ldots$$

$$m_{n+1}^0 = (0,0,\ldots,1)\cdot\boldsymbol{m}^0 = \boldsymbol{c}_{n+1}^0\cdot\boldsymbol{m}^0\,.$$

Next assume $\boldsymbol{c}_1^{i-1},\ldots,\boldsymbol{c}_{n+1}^{i-1}$ are vectors of integers of length $n+1$ such that

$$m_1^{i-1} = \boldsymbol{c}_1^{i-1}\cdot\boldsymbol{m}^0$$

$$\ldots$$

$$m_{n+1}^{i-1} = \boldsymbol{c}_{n+1}^{i-1}\cdot\boldsymbol{m}^0\,.$$

Then, by the definition of Q^i,

$$m_1^i = m_2^{i-1} - a_1^i m_1^{i-1}$$

$$\ldots$$

$$m_n^i = m_{n+1}^{i-1} - a_n^i m_1^{i-1}$$

$$m_{n+1}^i = m_1^{i-1}$$

and, by substitution,

$$m_1^i = (\boldsymbol{c}_2^{i-1} - a_1^i\boldsymbol{c}_1^{i-1})\cdot\boldsymbol{m}^0$$

$$\ldots$$

$$m_n^i = (\boldsymbol{c}_{n+1}^{i-1} - a_n^i\boldsymbol{c}_1^{i-1})\cdot\boldsymbol{m}^0$$

$$m_{n+1}^i = \boldsymbol{c}_1^{i-1}\cdot\boldsymbol{m}^0\,.$$

Therefore $\boldsymbol{c}_j^i$ can be defined in terms of $\boldsymbol{c}_1^{i-1},\ldots,\boldsymbol{c}_{n+1}^{i-1}$, and $\boldsymbol{a}^i$ so that no multiprecision arithmetic need be done until the sequence of correct a^i terminates. The argument can be repeated for the algorithm P and is omitted here. An earlier related result can be found in Blankenship [3].

This completes the discussion of the algorithms. Next, numerical results in the case $n=2$ will be presented.

3. Empirical Results.

In this section, the algorithms corresponding to P and Q will be studied empirically for $n=2$. The algorithm for P will be denoted as the Jacobi algorithm without order while that for Q will be denoted as the Jacobi algorithm with order. The large integer arithmetic was carried out on the Maniac II.

First, we consider the digit frequencies and invariant measures for P and Q. Table 1 gives the approximation to the density of the invariant measure corresponding to $T(Q)$ that was found in [2]. Similarly, the approximate density for $S(P)$ was found and is given in Table 2. Both of these results and the succeeding empirical results were obtained by iteration of the algorithms on the vector $\boldsymbol{m}=(m_1, m_2, m_3)$ where

Table 1. *Approximation of the density of the invariant measure for the Jacobi algorithm without order.*

x_2										
1.0	1.13	0.83	0.80	0.57	0.57	0.73	0.80	0.90	0.93	0.63
0.9	1.00	0.80	0.97	0.50	0.50	0.67	1.03	0.63	0.40	0.60
0.8	0.73	1.07	1.17	0.80	1.00	0.93	0.53	0.50	0.53	0.63
0.7	1.23	0.90	1.20	0.87	1.30	0.77	0.67	0.90	0.60	0.83
0.6	1.23	0.93	1.40	1.23	1.13	1.17	0.83	0.57	0.87	0.63
0.5	1.27	1.23	1.33	1.47	0.83	0.63	0.77	1.00	0.67	0.57
0.4	1.53	1.33	1.20	0.77	1.07	0.93	1.00	1.10	0.97	0.67
0.3	1.70	1.67	1.70	0.80	0.90	0.77	0.90	0.67	0.77	0.90
0.2	1.73	1.33	1.43	1.20	1.20	1.10	1.47	1.23	1.03	0.83
0.1	2.20	1.43	1.43	1.77	1.37	1.37	1.33	1.13	1.00	1.13
0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0 x_1

$$m_1 = \sum_{i=1}^{200} (5^{[2(200-i)+1]13} - [5^{[2(200-i)+1]13}/2^{43}])2^{43i} ,$$

$$m_2 = \sum_{i=1}^{200} (5^{26(200-i)} - [5^{26(200-i)}/2^{43}])2^{43i} ,$$

and

$$m_3 = 2^{8600} .$$

Each m_i requires 2589 decimal digits.

If $B_m(x)$ is the set of numbers with the same first m $\boldsymbol{a}$ as x, then the entropy of S or T ($h(S)$ or $h(T)$) satisfies

$$\lambda(B_m(x)) \sim e^{-mh(S)}$$

or

$$\lambda(B_m(x)) \sim e^{-mh(T)} ,$$

Table 2. *Approximation of the density of the invariant measure for the Jacobi algorithm with order. Numbers on the diagonal refer to the upper left triangular portion of the square.*

x_2 \ x_1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	1.50	1.70	1.73	2.03	1.87	1.70	1.63	1.63	1.57	1.20
0.9	1.70	1.77	1.87	1.53	1.37	1.60	1.43	1.83	1.27	
0.8	2.27	2.40	1.97	1.77	1.47	1.53	1.53	1.33		
0.7	2.33	1.67	2.37	1.47	2.03	2.07	1.67			
0.6	2.23	2.23	2.40	2.27	1.93	1.80				
0.5	2.07	2.67	2.43	2.43	2.00					
0.4	2.63	2.43	2.70	1.07						
0.3	2.90	2.70	2.60							
0.2	2.60	2.20								
0.1	2.93									

for almost all $\boldsymbol{x}$. It was found in [2] that

$$h(T) \cong 3.5\ .$$

Similarly, $h(S)$ was approximated by

$$h(S) \cong 4.9\ .$$

This is the first indication of the more rapid convergence of S. The area of $B_m(\boldsymbol{x})$ behaves like $e^{-3.5m}$ for T while it behaves like $e^{-4.9m}$ for S.

Next, digit frequencies are considered. Since $a=(a_1,a_2)$ with $0<a_1\leqq a_2$, the "size" of $\boldsymbol{a}$ is considered to be equivalent to a_2. Of course, the larger a_2 cause the algorithm to converge faster, but frequent occurrences of multiprecision a_2 would cause difficulties. In Table 3, this last possibility is seen not to occur. Again, the transformation S will tend to have larger digits than T and, thus, converge faster.

Table 3. *Digit frequencies for the first* 1000 $\boldsymbol{a}$ *for both Jacobi algorithms.*

Size of a_2	1	2	3	4	5	6	7	8	9
Frequency for T	.433	.164	.095	.064	.039	.029	.025	.018	.004
Frequency for S	.221	.173	.130	.070	.052	.057	.037	.031	.025

Size of a_2	10–99	100–999	1000–9999	$10^4-\infty$
Frequency for T	.115	.014	.000	.000
Frequency for S	.175	.023	.005	.001

Next, the two algorithms and some of their Lehmer modifications are compared. On the Maniac II, running times are easily available in seconds. It should be mentioned that

$$P^{5060}(\boldsymbol{m}) = Q^{3531}(\boldsymbol{m}) = (0,0,64)\ .$$

Therefore,

$$\gcd(\boldsymbol{m}) = 64\ .$$

The 5060 iterations of P took 420 seconds while the 3631 iterations of Q took 370 seconds.

For the Lehmer algorithms, after each stage at which multiprecision arithmetic was done, the procedure was to include L more decimal places than had been previously used. This made it unnecessary to locate the largest place in the transformed $\boldsymbol{m}$. The procedure was found to stabilize at a certain number of digits for the triangle approximation. This number of digits tended to be about $2L$. With $L=9$, 5000 P iterations took 169 seconds and 3600 Q iterations took 153 seconds.

Table 4 gives running times for Q, the Jacobi algorithm without order, without the Lehmer algorithm and with $L=9$ and $L=30$. Table 5 gives similar information for P.

Table 4. *Running times for the Jacobi algorithm without order.*

Number of Iterations	Digits Remaining	Time for Q	Time for Q, $L=9$	Time for Q, $L=30$
0	2589	0	0	0
500	2330	77	24	20
1000	2088	146	45	38
2000	1679	262	84	75
3000	1060	346	119	108

Table 5. *Running times for the Jacobi algorithm with order.*

Number of Iterations	Digits Remaining	Time for P	Time for P, $L=9$	Time for P, $L=30$
0	2589	0	0	0
500	2241	93	29	25
1000	1884	172	57	49
2000	1166	292	103	92
3000	459	357	136	130

It is an interesting question to determine how many single precision steps are performed during each Lehmer cycle. In each of the following cases, 500 iterations were performed. For the Jacobi algorithm without order (Q) and $L=9$, the mean number of steps per cycle, $\bar{x}$, was 16.7 with a standard deviation, s, of 4.21. For $L=30$, $\bar{x}=50$ and $s=14.88$. For the Jacobi algorithm with order (P) and $L=9$, $\bar{x}=12.5$ and $s=3.26$ while, for $L=30$, $\bar{x}=38.46$ and $s=10.11$. Clearly, the ordering operation causes P to have fewer single precision steps per Lehmer cycle.

Another study that can be performed is a search for the optional value of L for $\boldsymbol{m}$. For Q, the Jacobi algorithm without order, 1000 iterations were performed and timed for $L=5(1)50$. The time for $L=5$ was 58 seconds and the time for $L=9$ was 45 seconds. The minimum time for 38 seconds was at $L=30$. At $L=50$, the time was 51 seconds.

4. Conclusion.

Generally, the results of section 3 indicate that, for $n=2$, the Jacobi algorithm with order is superior to the Jacobi algorithm without order. However, the usual algorithm to compute the gcd of three integers uses Euclid's algorithm to proceed in a pairwise fashion:

$$\gcd(m_1, m_2, m_3) = \gcd(m_1, \gcd(m_2, m_3)).$$

We conjecture that, on conventional machines, the pairwise algorithm is faster than any Jacobi algorithm. Unfortunately, no such study, theoretical or empirical, has yet appeared. However, it is possible that on vector machines, such as the CRAY-1 or the CDC STAR, the Jacobi algorithm be faster than the pairwise algorithm.

It is interesting that examples can be found where the Jacobi algorithm with order takes more iterations than the Jacobi algorithm without order. This observation is due to R. W. Johnson who found the example of $\boldsymbol{m}=(1396, 7694, 8593)$. The Jacobi algorithm with order takes 6 steps to find $\gcd(\boldsymbol{m})=1$ while the Jacobi algorithm without order takes 5 steps. If it is possible to characterize when this unexpected occurrence takes place, then it might be possible to devise an algorithm which alternates between ordered and unordered steps, and which would converge faster than P or Q.

Along these lines, Kronecker [8] has shown that the least remainder algorithm never takes more iterations than any other Euclidean algorithm. Is there a corresponding result for the Jacobi algorithm?

Also, Lame's theorem [8] gives the maximum number of divisions required to find the greatest common divisor using the Euclidean algorithm. The number of iterations does not exceed five times the number of digits in the smallest integer. Such a result for the Jacobi algorithm would involve some generalization of Fibonacci numbers which would be the worst case.

ACKNOWLEDGEMENT. The author thanks Dr. Roger W. Johnson of Lewistown College Center for many helpful discussions and for the example in the conclusion.

REFERENCES

1. L. Bernstein, *The Jacobi–Perron algorithm. Its theory and application*, Lecture Notes in Mathematics 207, Springer-Verlag, Berlin, Heidelberg, New York, 1970.
2. W. A. Beyer and M. S. Waterman, *Ergodic computations with continued fractions and Jacobi's algorithm*, Numer. Math. 19 (1972), 195–205. Errata 20 (1973), 430.
3. W. A. Blankenship, *A new version of the Euclidean algorithm*, Amer. Math. Monthly 70 (1963), 742–745.
4. B. F. Caviness, *A Lehmer-type greatest common divisor algorithm for Gaussian integers*, A talk presented to the SIAM-SIGNUM meeting, Austin, Texas, October 18, 1972.
5. D. E. Knuth, *The Art of Computer Programming*, Volume 2 / Seminumerical Algorithms, Addison–Wesley, Massachusetts, California, London, 1969.
6. D. H. Lehmer, *Euclid's algorithm for large numbers*, Amer. Math. Monthly 45 (1938), 227–233.
7. F. Schweiger, *The metrical theory of Jacobi–Perron algorithm*, Lecture Notes in Mathematics 334, Springer-Verlag, Berlin, Heidelberg, New York, 1973.
8. J.V. Uspensky and M. A. Heaslet, *Elementary Number Theory*, McGraw–Hill, New York, London, 1939.
9. M. S. Waterman, *On F-expansions of rationals*, Aequationes Math. 13 (1975), 263–268.
10. M. S. Waterman, *A Jacobi algorithm and metric theory for greatest common divisors*, J. Math. Anal. Appl. 59 (1977), 288–300.

LOS ALAMOS SCIENTIFIC LABORATORY
LOS ALAMOS, NEW MEXICO 87545
U.S.A.