Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction

Michael Monagan*
Department of Mathematics, Simon Fraser University,
Burnaby, B.C., V5A 1S6, Canada.

Abstract

Let $n/d \in \mathbb{Q}$, m be a positive integer and let $u = n/d \bmod m$. Thus u is the image of a rational number modulo m. The rational reconstruction problem is; given u and m find n/d. A solution was first given by Wang in 1981. Wang's algorithm outputs n/d when $m > 2M^2$ where $M = \max(|n|, d)$. Because of the wide application of this algorithm in computer algebra, several authors have investigated its practical efficiency and asymptotic time complexity.

In this paper we present a new solution which is almost optimal in the following sense; with controllable high probability, our algorithm will output n/d when m is only a few bits longer then $2|n|d$. This means that in a modular algorithm where m is a product of primes, the modular algorithm will need about one prime more than the minimum necessary. Thus if $|n| \ll d$ or $d \ll |n|$ the new algorithm saves up to half the number of primes. The new algorithm is designed to have the additional desirable property that it will fail with high probability when $m < 2|n|d$.

1 Introduction

Rational reconstruction, originally developed by Wang in [14] to recover a partial fraction decomposition of a rational function in $\mathbb{Q}(x)$ from it's image modulo $m = p^k$, a prime power, has become an important and very useful tool for the development of efficient algorithms in computer algebra. It enables modular algorithms to recover rational numbers from their

*Supported by NSERC of Canada and the MITACS NCE of Canada.

images modulo a large integer modulus m, usually a prime power or product of machine primes. For example, Encarnacion's modular GCD algorithm in [2] computes the monic gcd g of two univariate polynomials f_1, f_2 over an algebraic number field $\mathbb{Q}(\alpha)$ by computing the $\gcd(f_1, f_2)$ modulo $m = p_1 \times p_2 \times ... \times p_k$, a product of machine primes. It does this by computing the gcd modulo each prime and applying the Chinese remainder theorem. The rational coefficients of g are then recovered from their images modulo m using rational reconstruction. The advantage of Encarnacion's algorithm over the modular GCD algorithm of Langemyr and MacCallum in [9] is that it is *output sensitive;* that is, the number of primes needed depends on the size of the rational coefficients in the output g and not on bounds based on the size of the inputs f_1, f_2 and the minimal polynomial for α which may be much too large. A second advantage is that rational reconstruction does not require that we know a priori a multiple of the LCM of the denominators of the rational coefficients appearing in g as is the case in Langemyr and MacCallum's modular GCD algorithm. Some other applications where rational reconstruction is used, central to computer algebra, include solving linear systems over $\mathbb{Q}$ (see [12]), early detection of factors in the Berlekamp-Hensel procedure (see [16]), and Gröbner basis computation over $\mathbb{Q}$ (see [7]).

Let $n, d \in \mathbb{Z}$ with $d > 0$ and $\gcd(n, d) = 1$. Let m be a positive integer satisfying $\gcd(m, d) = 1$. Let $u = n/d \bmod m$. The rational reconstruction problem is: given u and m find n and d. One aspect of the problem is how large m has to be before the algorithm will output n/d. It is not hard to see that if the algorithm is to recover all rational numbers with numerators $\leq |n|$ and denominators $\leq d$, then the modulus $m > 2|n|d$. A solution to this problem without proof was first given by Wang in [14]. Wang's algorithm, a simple modification of the (extended) Euclidean algorithm, outputs n/d provided $|n|$ and d are both less than $\sqrt{m/2}$, that is, $m > 2\max(|n|, d)^2$. A proof was subsequently given by Wang, Guy and Davenport in [15]. In the following we develop an example to motivate the design of a more efficient solution.

Let $p_1, p_2, p_3, ...$ be a sequence of near constant length primes being used by a modular algorithm such as Encarnacion's modular GCD algorithm in [2]. Suppose we wish to recover the polynomial

$$g \;=\; x^2 + \frac{123456789}{5}\,x - \frac{4115}{2263}\,\alpha + 12345654321$$

from it's image u modulo $m = m_k$ where $m_k = p_1 \times p_2 \times ... \times m_k$. In the modular GCD algorithm, because we do not know in advance how large the coefficients of g are, we design the modular algorithm to attempt re-

covery of g after we have computed g modulo each prime so that it can terminate as soon as possible. Suppose, for example, we use 3 digit primes $997, 991, 983, ...$ How many primes will we need to recover g in our example? Wang's algorithm will recover the coefficients of g when

$$m = m_k > 2 \max_{n/d \in g} (|n|, d)^2,$$

which depends on the largest integer appearing in the rational coefficients of g, namely, 12345654321. Wang's algorithm needs *seven* 3 digits primes on this example, namely, $997, 991, 983, 977, 971, 967, 953$. However, the rationals in g are not uniform in shape which is quite typical for real examples that we have encountered. Our new algorithm will require that m be a few bits longer than $\max_{n/d \in g}(|n|d) = 12345654321$; it needs only *five* 3 digit primes on this example.

Let j be the smallest integer such that $m = m_j = p_1 \times ... \times p_j > 2|n|d$. Thus m_j is the smallest modulus for which rational reconstruction could recover g. The main result of our paper is a new algorithm for rational reconstruction with the following properties:

(i) it will succeed with high probability for all moduli $m = m_k$ with $k > j$, thus, we need approximately one more prime than the minimum possible, and

(ii) it will fail with high probability for all moduli $m = m_k$ with $k < j$ primes.

For Encarnacion's modular GCD algorithm (i) means a reduction of the number of primes needed to reconstruct g by up to a factor of 2, and (ii) means that rational reconstruction will fail early when m is too small thus avoiding most of the trial divisions in Encarnacion's algorithm.

Outline of Paper

In section 2 we explain Wang's rational reconstruction algorithm and we reference subsequent work by several authors to accelerate Wang's algorithm. In section 3 we present our new algorithm. We prove that our algorithm always outputs n/d when $m > 9d^2n^2$ and that it outputs n/d with high probability when $m > 2^k d|n| \log_2 m$ with $k = 20$. In the conclusion, we make some remarks about the implementation of the algorithm and briefly outline the modifications needed for our algorithm so that it may be applied to solve the related problem of rational function reconstruction in one parameter t over a finite field which is needed to recover parameters in modular algorithms.

2 Rational Reconstruction

As in the introduction let $n, d \in \mathbb{Z}$ with $d > 0$ and $\gcd(n, d) = 1$. Let m be a positive integer satisfying $\gcd(m, d) = 1$ and let $u = n/d \bmod m$. Thus n/d is the rational we are trying to reconstruct from it's image u modulo m. To explain how Wang's rational reconstruction algorithm and our new algorithm work we need to refer to the integers appearing in the (extended) Euclidean algorithm. Let $\lfloor a/b \rfloor$ denote the integer quotient of a divided by b. On input of m and u the Euclidean algorithm computes integers r_i, s_i, t_i for $i = 0, 1, ..., n, n+1$ and quotients q_{i+1} for $i = 1, ..., n$ as follows:

Euclidean algorithm.
Input: integers $m > u \geq 0$.

Set $(r_0, s_0, t_0) = (m, 1, 0)$.
Set $(r_1, s_1, t_1) = (u, 0, 1)$.
Set $n = 0, k = 1$.
While $r_k \neq 0$ do
  Set $q_{k+1} = \lfloor r_{k-1}/r_k \rfloor$
  Set $r_{k+1} = r_{k-1} - q_{k+1} r_k$.
  Set $s_{k+1} = s_{k-1} - q_{k+1} s_k$.
  Set $t_{k+1} = t_{k-1} - q_{k+1} t_k$.
  Set $n = k, k = k + 1$.

Let $g = \gcd(m, u)$. At the end of the Euclidean algorithm we have

(i) $r_0 > r_1 > ... > r_n = g > r_{n+1} = 0$, and,

(ii) $t_0 < t_1 \leq |t_2| < ... < |t_n| < |t_{n+1}| = m/g$, and

(iii) $s_i m + t_i u = r_i$ for $i = 0, 1, ..., n + 1$.

where (ii) and (iii) follow by induction on n. Solving (iii) for u modulo m there are two cases to consider depending on whether $\gcd(m, t_i) = 1$ or not.

Case 1: $\gcd(m, t_i) = 1$: we have $r_i/t_i \equiv u \bmod m$.

Case 2: $\gcd(m, t_i) > 1$: let $d_i = \gcd(m, t_i)$. We have $d_i | m$ and $d_i | t_i$ and $s_i m + t_i u = r_i$ implies $d_i | r_i$, and hence, $(r_i/d)/(t_i/d) \equiv u \bmod (m/d)$.

In a modular algorithm where m is a single large prime or a product of machine primes or a power of a machine prime, where by machine prime we mean a 30 bit prime on a 32 bit machine, case 2 will be exceptional. Hence,

the Euclidean algorithm generates a sequence of rationals $r_1/t_1, ..., r_n/t_n$ which are candidates for the solution of the rational reconstruction problem. Notice that $t_1 = 1, r_1 = u$ is always a solution; it is the integer solution to the rational reconstruction problem. We state the following result from [15] which allows us to "select the right solution" for sufficiently large m.

Theorem 1 *Wang, Guy, Davenport, 1982. Let $n, d \in \mathbb{Z}$ with $d > 0$ and $\gcd(n, d) = 1$. Let $m \in \mathbb{Z}$ with $m > 0$ and $\gcd(m, d) = 1$. Let $u = n/d \bmod m$. Let $N, D \in \mathbb{Z}$ such that $N \geq n$ and $D \geq d$. Then*

(i) if $m > 2ND$ the modular map $\phi : \mathbb{Q} \to \mathbb{Z}_m$ given by $\phi(x) = x \bmod m$ is injective, that is, for any $0 \leq u < m$ there is at most one rational number $n/d \equiv u \bmod m$, and,

(ii) if $m > 2ND$ and $u \equiv n/d \bmod m$ then on input of m and u there exists a unique index i in the Euclidean algorithm such that $s_i m + nu = d$. Moreover, i is the first index such that $r_i \leq N$.

We refer the reader to von zur Gathen and Gerhard [3] for a more accessible reference for the proof of this result. The algorithm follows.

Algorithm Rational Reconstruction (RR), Wang, 1981.
Input: Integers $m, N, D > 0, u \geq 0$ s.t. $m > u$ and $2ND < m$.
Output: Either $n, d \in \mathbb{Z}$ s.t. $|n| \leq N$, $0 < d \leq D$, $\gcd(n, d) = 1$, and $n/d \equiv u \bmod m$, or FAIL implying no such rational n/d exists.

Set $(r_0, t_0) = (m, 0)$.
Set $(r_1, t_1) = (u, 1)$.
While $r_1 > N$ do
  Set $q = \lfloor r_0/r_1 \rfloor$
  Set $(r_0, r_1) = (r_1, r_0 - q\, r_1)$.
  Set $(t_0, t_1) = (t_1, t_0 - q\, t_1)$.
Set $(n, d) = (r_1, t_1)$.
If $d < 0$ then set $(n, d) = (-n, -d)$.
If $d \leq D$ and $\gcd(n, d) = 1$ then output (n, d).
Otherwise output FAIL.

Remark on the gcd condition: The requirement that $\gcd(n, d) = 1$ is not present in in the original description of the algorithm in [14] and [15]. In [1], Collins and Encarnacion pointed out that because the original algorithm fails to establish that $\gcd(d, m) = 1$, if case 2 above occurs, the original algorithm can output an invalid result. For example, if we take

$m = 12, u = 5, N = 2, D = 2$ we find the relation $2m - 2u = 2$ and the bad output would be $n = -2, d = 2$ which does not satisfy $n/d \equiv u \bmod m$. Hence case 2 must be detected. Collins and Encarnacion also point out that since $\gcd(n, d) = 1$ iff $\gcd(d, m) = 1$, and $m > d$, it is more efficient to make the test $\gcd(n, d) = 1$.

In Wang's original presentation of the algorithm he sets $N = D = \lfloor\sqrt{m/2}\rfloor$ where m is assumed to be odd. Thus the algorithm will output n/d for all $m > 2\max(|n|, d)^2$. Thus as the modulus m grows in the modular GCD algorithm, each time rational reconstruction is attempted, half of the bits of m are used to reconstruct the numerator and half are used to reconstruct the denominator. For example, consider $m = 13$. Setting $N = D = \lfloor\sqrt{6}\rfloor = 2$ algorithm RR recovers $\{0/1, 1/1, -1/2, 2/1, -2/1, 1/2, -1/2\}$ from their images modulo 13. Running the algorithm for $m = 13$ on $0 < u < m$ with several allowable settings for N, D we obtain the data in Table 1.

N, D	$u = 1$	2	3	4	5	6	7	8	9	10	11	12
$2, 2$	1	2	F	F	F	$-\frac{1}{2}$	$\frac{1}{2}$	F	F	F	-2	-1
$3, 2$	1	2	3	F	$-\frac{3}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$	F	-3	-2	-1
$2, 3$	1	2	F	$-\frac{1}{3}$	$\frac{2}{3}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{2}{3}$	$\frac{1}{3}$	F	-2	-1

Table 1: Rational Reconstruction (F indicates failure)

Suppose u is chosen uniformly at random on $0 \leq u < m$. One can show that if we set $N = D = \sqrt{m/2}$ then the probability that algorithm RR succeeds approaches $6/\pi^2 = 0.6079$ as m increases. This means that when we attempt to reconstruct the rational coefficients of a polynomial and m is not yet large enough, algorithm RR will successfully reconstruct $1/(1-6/\pi^2) = 2.55$ coefficients on average before it fails. More significantly, when it succeeds in reconstructing all the coefficients for the first time, there is a significant probability that we do not have the right answer yet. One way to ensure that the output from algorithm RR is right with high probability is to select N and D such that the $2ND$ is significantly smaller than m. For example, in Magma (versions 2.8, 2.9, and 2.10), Steel [13] does the following for his implementation of Encarnacion's modular GCD algorithm; he tries $N = m/2^4, D = 1$ and then $N = \sqrt{m}/2, D = m/N/2^4$. The reason for the first setting is that it means that if g is monic with integer coefficients, a common case, half the number of primes are needed.

2.1 Complexity of Rational Reconstruction

The asymptotic time complexity of the classical (extended) Euclidean algorithm, and therefore also of algorithm RR, is $O(\log^2 m)$. The (extended) Euclidean algorithm may be accelerated by a constant factor using Lehmer's GCD algorithm (see Knuth [8]). A further factor of 2 improvement on Lehmer's GCD algorithm is obtained by Jebelean in [6]. In [1], Collins and Encarnacion show how to modify both Lehmer's algorithm and Jebelean's algorithm for rational reconstruction. For a modulus m of 1550 bits in length (437 decimal digits), the speedup they obtained (see Table 1 in [1]) using Lehmer's algorithm instead of the ordinary (extended) Euclidean algorithm was a factor of 3.6 and a further factor of 2 speedup using Jebelean's algorithm was obtained.

An asymptotically fast version of rational reconstruction, with the same complexity as the asymptotically fast integer GCD algorithm of Schönhage, is described by Pan and Wang in [12]. Thus the theoretical bit complexity of rational reconstruction is $O(n \log^2 n \log\log n)$ where $n = \log m$ here. The authors did not implement their algorithm and remarked that it may not be practical. However, previously, in unpublished work, Allan Steel [13] implemented in Magma a fast rational reconstruction algorithm based on the fast half-gcd algorithm for $F[x]$ in Peter Montgomery's PhD thesis [11]. The algorithm is effective in practice for large m. For m below 500 bits in length, Magma uses Lehmer's algorithm (see Knuth [8]). At 500 bits, Steel's Magma implementation switches to the fast rational reconstruction algorithm where Karatsuba's algorithm is being used for integer multiplication. An FFT multiplication kicks in at 20,000 bits. We demonstrate the effectiveness of this implementation in the following experiment. The following Maple session constructs first a 50,000 digit modulus m, an integer B satisfying $2B^2 < m$, a positive fraction n/d satisfying $n < B$ and $d < B$, and the image $u = n/d \bmod m$ and the integer c.

```
> p,q,r := 10^10+19, 10^10-33, 10^10-57;
> k := 5000: m := p^k: B := isqrt(iquo(m,2));
> n := q^(k/2) mod B: d := r^(k/2) mod B: u := n/d mod m:
> c := u*m:
```

The table below includes timings, in CPU seconds, comparing Magma 2.10 and Maple 9 on various arithmetic computations where RR denotes rational reconstruction. Maple 9 is using the GNU multiple precision integer arithmetic package [4], which, like Magma, includes an FFT based integer multiplication routine for large integer multiplication, but, unlike Magma

does not include an asymptotically fast Euclidean algorithm. The table of timings demonstrate clearly the effectiveness of the accelerated Euclidean algorithm in Magma and its application to computing gcds in $\mathbb{Z}$, inverses in $\mathbb{Z}_m$, and rational reconstruction (RR).

	k	$u \times m$	$c \,/\, m$	$\gcd(u, m)$	$u^{-1} \bmod m$	$RR(m, u)$
Magma	5000	0.03s	0.09s	0.41s	1.26s	1.1s
2.10	10000	0.06s	0.25s	1.22s	3.63s	3.2s
	20000	0.15s	0.64s	3.41s	9.80s	9.0s
Maple	5000	0.01s	0.04s	0.27s	5.91s	6.2s
9	10000	0.03s	0.10s	1.07s	17.70s	24.0s
	20000	0.08s	0.25s	4.25s	70.40s	97.7s

Remark: The timing data for Magma 2.10 suggests that Karatsuba multiplication is being used in this range for the latter three operations since the timings increase consistently by a factor of 3. The data confirms that integer gcd in Maple 9 is $O(\log^2 m)$.

3 Maximal Quotient Rational Reconstruction

In practice we have observed that for the modular GCD algorithm over a number field $\mathbb{Q}(\alpha)$, the rational coefficients of polynomials often have much smaller denominators than numerators. One reason for this is that if the minimal polynomial for α is monic over $\mathbb{Z}$, as is often the case, it only contributes to the growth of numerators in $\mathbb{Q}(\alpha)$ when we multiply elements of $\mathbb{Q}(\alpha)$. This observation about the relative size of numerators and denominators is not specific to polynomials over number fields; it is most often that case that rational coefficients have different sizes in real problems.

On such problems rational reconstruction with $N = D = \sqrt{m/2}$ will require up to twice as many primes as are necessary. A practical improvement that we tried was to allocate 2/3 of the digits of m to the numerators and 1/3 to the denominators, i.e., we used $D = \lfloor \sqrt[3]{m} \rfloor$ and $N = \lfloor m/(2D^2) \rfloor$. The problem with this is that we may be wrong. We usually just don't know in advance what the best choice for N and D is. We say that the "shape" of the rationals is unknown. But it turns out that this does not matter for we can modify the Euclidean algorithm so that it will work with high probability regardless of the shape of the rational numbers. We first make an observation about the Euclidean algorithm relating the size of the quotient q_{i+1} to the size of the rational r_i/t_i and the modulus m.

Lemma 1 $\quad m/3 < q_{i+1}|t_i|r_i \leq m \quad$ for $\;\; i = 1, 2, ..., n.$

Proof: The proof of Lemma 1 follows from the following known property of the Euclidean algorithm which may be established by induction on i:

$$(-1)^i m = r_i t_{i-1} - r_{i-1} t_i \quad \text{for} \;\; i = 1, 2, ..., n.$$

Let U_i denote the quantity $q_{i+1}\,|t_i|\,r_i/m$. We need to show $1/3 < U_i \leq 1$. Substituting for m we have

$$U_i = \frac{|q_{i+1} t_i r_i|}{|r_{i-1} t_i - r_i t_{i-1}|}.$$

Dividing through by r_i and t_i we have

$$U_i = \frac{q_{i+1}}{|r_{i-1}/r_i - t_{i-1}/t_i|}.$$

Now $0 \leq |t_{i-1}| < |t_i|$ and t_{i-1} has opposite sign to t_i which imply $0 \leq -t_{i-1}/t_i < 1$. Hence we have

$$\frac{q_{i+1}}{1 + r_{i-1}/r_i} < U_i \leq \frac{q_{i+1}}{r_{i-1}/r_i}.$$

Now substituting for r_{i-1}/r_i from $r_{i+1} = r_{i-1} - q_{i+1} r_i$ we have

$$\frac{q_{i+1}}{q_{i+1} + r_{i+1}/r_i + 1} < U_i \leq \frac{q_{i+1}}{q_{i+1} + r_{i+1}/r_i}.$$

Since $0 \leq r_{i+1} < r_i \implies 0 \leq r_{i+1}/r_i < 1$ we have

$$\frac{q_{i+1}}{q_{i+1} + 2} < U_i \leq \frac{q_{i+1}}{q_{i+1} + 0} = 1.$$

The left-hand-side is minimized when $q_{i+1} = 1$ which gives us $1/3 < U_i \leq 1$ as required.

Remark: For input m, u with $u = 1$ we have at step $i = 1$, $q_{i+1}|t_i|r_i = m/1 \times 1 \times 1 = m$ and hence the right hand side of the inequality in Lemma 1 is tight. For the left-hand-side of the inequality we have computed the smallest ratio $U_i = q_{i+1}|t_i|r_i/m$ appearing in the Euclidean algorithm for each modulus $1 < m < 10^5$ over all inputs $0 < u < m$. The data shows that the smallest ratio approaches $1/3$, from above, linearly, as m increases.

The inequality on the left of Lemma 1 tells us that if r_i/t_i is small relative to m, that is, $r_i|t_i| \ll m$, that is, our algorithm should be able to recover this rational from $u = r_i/t_i \bmod m$, then the quotient q_{i+1} computed in the next step of the Euclidean algorithm is necessarily large, namely, $q_{i+1} > m/|3t_i r_i|$. The following table tabulates r_i, t_i, q_{i+1} and the quantity $1/3 < U_i \leq 1$ appearing in the proof of Lemma 1 for a run of the Euclidean algorithm with input $m = 10^6 - 17$ and $u = 137613$.

i	r_i	t_i	q_{i+1}	U_i
1	137613	1	7	.9633
2	36692	-7	3	.7705
3	27537	22	1	.6058
4	9155	-29	3	.7965
5	72	109	127	.9967
6	11	-13872	6	.9156
7	6	83341	1	.5001
8	5	-97213	1	.4861
9	1	180554	5	.9028

Table 2: Rationals in the Euclidean algorithm

The quotients are small, which is typical for the Euclidean algorithm except one, namely $q_5 = 127$, which corresponds (by Lemma 1) to the only small fraction, 72/109. This observation suggests that rational reconstruction simply output the rational r_i/t_i for which q_{i+1} is the maximal quotient. We will refer to this algorithm as Maximal Quotient Rational Reconstruction (MQRR). Can there be more than one large quotient? The following lemma tells us that there can only be one large quotient if m is large enough.

Lemma 2 *Let $n, d \in \mathbb{Z}$ with $d > 0$ and $\gcd(n, d) = 1$. Let $m \in \mathbb{Z}$ with $m > 0$ and $\gcd(m, d) = 1$. Let $u = n/d \bmod m$ and let i be an index with q_{i+1} a maximal quotient in the Euclidean algorithm when given input (m, u). Thus $u \equiv r_i/t_i \bmod m$. If $|n|d < \sqrt{m}/3$ then i is unique and $r_i/t_i = n/d$.*

Proof: Let $N = |n|$ and $D = d$. Since $3|n|d < \sqrt{m}$ implies $2|n|d = 2ND < m$, then from Theorem 1 there exists a unique index i in the Euclidean algorithm with input (m, u) such that $r_i/t_i = n/d$. Let $q = q_{i+1}$ be the next quotient. Then inequality (1) implies $m/3 < q|n|d$ which implies $q > m/(3|n|d)$. But $\sqrt{m} > 3|n|d$ which implies $q > \sqrt{m}$. Now the product of

the quotients of the Euclidean algorithm also satisfies $\Pi_{i=1}^{n} q_{i+1} \leq m$, hence, if $q > \sqrt{m}$ it is the largest quotient. Therefore the index i of the maximal quotient implies $r_i/t_i = n/d$.

Remark: For $m = 10^{20} - 1$ and $u = 10^{10}$ there are two distinct rationals $r_1/t_1 = 10^{10}$ and $r_3/t_3 = 10^{-10}$ for which the corresponding quotients $q_2 = 10^{10} - 1$ and $q_4 = 10^{10} - 1$ are both large. Since they are very close to $\sqrt{m}$ in length, the statement of lemma 2 is tight.

Lemma 2 says that for n/d fixed algorithm MQRR will output n/d provided m is large enough, that is, when $m > 9n^2d^2$. This guarantees that as the modulus m is increasing, algorithm MQRR will eventually succeed but it requires that the modulus m be more than twice as long as the length of nd before it must succeed. This means that algorithm MQRR yields no improvement over Wang's rational reconstruction algorithm in the worst case. Fortunately, however, this is not what happens in the average case. In fact, the new algorithm will output n/d with high probability when the length of the modulus m is only a modest number of bits longer than the length of nd. The basic reason for this is that large quotients in the Euclidean algorithm are rare. This motivates the following design for an improved rational reconstruction algorithm in which the input parameter T gives the user control over the probability that the algorithm will succeed on random input.

Algorithm MQRR (Maximal Quotient Rational Reconstruction)
Input: Integers $m > u \geq 0$ and $T > 0$.
Output: Either $n, d \in \mathbb{Z}$ s.t. $d > 0$, $\gcd(n, d) = 1$, $n/d \equiv u \bmod m$, and $T|n|d < m$, or FAIL.

If $u = 0$ then if $m > T$ then output 0 else output FAIL.
Set $(n, d) = (0, 0)$.
Set $(t_0, r_0) = (0, m)$.
Set $(t_1, r_1) = (1, u)$.
While $r_1 \neq 0$ and $r_0 > T$ do
    Set $q = \lfloor r_0/r_1 \rfloor$.
    If $q > T$ then set $(n, d, T) = (r_1, t_1, q)$.
    Set $(r_0, r_1) = (r_1, r_0 - q\, r_1)$.
    Set $(t_0, t_1) = (t_1, t_0 - q\, t_1)$.
end while.
If $d \neq 0$ and $\gcd(n, d) = 1$ then output n/d else output FAIL.

We wish now to determine a good value for T such that algorithm MQRR works with high probability, that is, if $m > |a|\, bT$ and $\gcd(b, m) = 1$ the algorithm on input of m and $u = a/b \bmod m$ will output a/b with high probability. To do this we need to study the distribution of the largest quotient that appears in the Euclidean algorithm. Let $Q = \max_{i=1}^{n} q_{i+1}$ denote the largest quotient of the Euclidean algorithm with input $m > u \geq 0$. We have made the following experiment using $m = 2^{64} - 59$ the largest 64 bit prime and using $m = 2^{128} - 159$ the largest 128 bit prime. We computed Q for 10^8 random inputs u on $[0, m)$ and tabulated, in Table 3, f_k the number of largest quotients in the range $[2^{k-1}, 2^k)$ that is, the number of quotients of length k bits. Also tabulated is the ratio $r_k = f_{k-1}/f_k$.

The data shows very clearly that the probability of obtaining a large quotient drops off exponentially by a factor of 2 after approximately the median value. For $m = 2^{64} - 59$, using linear interpolation for the interval $2^6 \leq Q < 2^7$ we find that the median maximal quotient is approximately $2^{6.276} = 77.5$. For $m = 2^{128} - 159$, using linear interpolation for the interval $2^7 \leq Q < 2^8$ we approximate the median maximal quotient with $2^{7.289} = 156$.

This means that for $m = 2^{64}$, if the parameter T is set to 77, algorithm MQRR will, with probability 1/2, succeed on random input. If we are using 30 bit primes in the modular GCD algorithm and we set $T = 2^{30}$ corresponding to one prime more than the minimum necessary to reconstruct, then the probability of obtaining a maximal quotient longer than 30 bits is less than one in ten million.

However, we must not fix T to be 2^{30} because the median value of Q *increases* as m increases. Thus the value of T used by algorithm MQRR must increase as m increases. We seek a result of the following form: given an error tolerance $0 < \epsilon = 2^{-k} \ll 1$, find q as a function of m such that $Pr(Q \geq q) = \epsilon$. This is too difficult to obtain. Instead we obtain the weaker result, sufficient for our purposes, based on the median value of Q.

Lemma 3 *Let Q be the maximal quotient of the Euclidean algorithm for random input u on $[0, m)$. Then*

$$Pr(Q \leq q) = 0.5 \quad \Longrightarrow \quad \log_2 q = \log_2 \log_2 m + O(1).$$

Thus, together with the observation that the beyond the median value of Q the ratios r_k are close to 2 means that we should set $\log_2 T = \log_2 \log_2 m + c$ for some constant c, say $c = 20$ or $c = 30$.

$m = 2^{64} - 59$			$m = 2^{128} - 159$		
k	f_k	r_k	k	f_k	r_k
3	27064	.0117	3	5	.0011
4	2317747	.1545	4	46496	.0174
5	15006123	.5736	5	2672302	.1770
6	26163579	1.113	6	15100610	.5920
7	23499297	1.531	7	25507027	1.104
8	15346618	1.782	8	23097400	1.505
9	8613460	1.912	9	15349534	1.750
10	4506037	1.976	10	8772209	1.881
11	2280491	2.006	11	4662860	1.952
12	1136600	2.027	12	2389109	1.983
13	560838	2.032	13	1204725	1.998
14	276037	2.040	14	602994	2.012
15	135283	2.031	15	299765	2.019
16	66600	2.033	16	148508	2.015
17	32757	2.030	17	73717	2.018
18	16140	2.105	18	36532	2.003
19	7669	1.988	19	18241	2.025
20	3858	2.002	20	9007	1.981
21	1927	2.127	21	4547	2.100
22	906	1.801	22	2165	1.935
23	503	2.140	23	1119	1.929
24	235	2.061	24	580	2.014
25	114	2.235	25	288	2.087
26	51	1.594	26	138	2.421
27	32	1.882	27	57	2.478
28	17	1.889	28	23	1.533
29	9	1.800	29	15	.9375
30	5	5.	30	16	3.200
31	1	1.	31	5	1.250
32	1	–	32	4	4.000
			34	1	–.

Table 3: Largest quotients in the Euclidean algorithm.

3.1 The Maximal Quotient

We investigate the distribution of the largest quotient appearing in the Euclidean algorithm to prove Lemma 3. If one assumes in the Euclidean algorithm that the remainder r_{i+1} of r_{i-1} divided by r_i is randomly distributed on $[0, r_i)$ then

$$Pr(q_{i+1} = q) \ = f(q) = 1/q - 1/(q+1), \quad \text{and}$$

$$Pr(q_{i+1} \leq q) \ = F(q) = \sum_{k=1}^{q} f(q) = 1 - 1/(q+1).$$

The probabilities $f(q)$ underestimate the size of the quotients that appear, in particular, $f(1)$ is too big. A more accurate approximation for the probability that a quotient q_{i+1} in the Euclidean algorithm is q is studied by Knuth in [8], section 4.5.3, namely

$$Pr(q_{i+1} = q) \ = g(q) = \log_2 \left[1 + \frac{1}{q(q+2)} \right]$$

and hence

$$Pr(q_{i+1} \leq q) \ = G(q) = \sum_{k=1}^{q} g(q) = 1 - \log_2 \frac{q+2}{q+1}.$$

Some values for $f(q)$ and $g(q)$ and their cumulative probabilities are tabulated below. We remark that $g(q)$ is accurate except when q is very large, e.g., $q = m$. Let n be the number of steps in the Euclidean algorithm and

q	1	2	3	4	5	8	16	32
$f(q)$	.500	.167	.083	.050	.033	.014	.0037	.00095
$g(q)$	.415	.170	.093	.059	.041	.018	.0050	.00130
$F(q)$	.500	.667	.750	.800	.833	.889	.9412	.96970
$G(q)$	.415	.585	.678	.737	.778	.848	.9175	.95693

let $Q = \max_{i=1}^{n} q_{i+1}$ be the largest quotient. If n is independent of Q then

$$Pr(Q \leq q) = Pr(q_{i+1} \leq q)^n.$$

The independence assumption is not true, however, because larger quotients will correlate with smaller n. If $m \gg Q$ the assumption is approximately

true. In which case we have the approximation

$$Pr(Q \leq q) \approx G(q)^n = (1 - \log_2 \frac{q+2}{q+1})^n.$$

We need also an approximation for n. For a given value of m we may run the Euclidean algorithm on random inputs $u \in [0, m)$ and use the average number of steps as an estimate for n. From experiment we observe that this approximation for n is quite stable. For $m = 2^{64} - 59$ we obtained $n = 37.8$ after 10^6 values of u. A theoretical estimate[1] for n is given in Knuth [8], namely,

$$n = 12 \ln 2/\pi^2 \ln m + C + ... = 0.8428 \ln m + 0.4671 + ...$$

Using this formula for n we find that the average number of steps of the Euclidean algorithm for input m, u where $m = 2^{64} - 59$ and u is chosen at random from $[0, m-1)$ is $n = 37.85$. This agrees with our value above from experiment. We now estimate the median largest quotient as a function of m, that is, the value of q such that

$$\Pr(Q \leq q) = G(q)^n = 1/2.$$

Taking logarithms of both sides we have $\log_2 G(q) = -1/n$ from which we obtain

$$\log_2 \frac{q+2}{q+1} = 1 - 2^{-1/n}.$$

Since m is large then n is also large and hence $1 - 2^{-1/n}$ is approximated well by $\ln 2/n$. Similarly, since q is large we approximate $\log_2 \frac{q+2}{q+1}$ with $1/(q \ln 2)$. Substituting for these approximations we have

$$1/(q \ln 2) \approx \ln 2/n \implies q \approx n/\ln^2 2.$$

Substituting n for $12 \ln 2/\pi^2 \ln m$ we obtain

$$q \approx 12/\ln 2/\pi^2 \ln m + ... = 12/\pi^2 \log_2 m + ...$$

which gives us the result that we wanted, namely,

$$\log_2 q \approx \log_2 \log_2 m + 0.2820 + ...$$

Using this approximation we can now estimate the median largest quotient for $m = 2^{64} - 59$. We obtain $\log_2 q = 6.282$ which is in very good agreement with the observed value of 6.276. For $m = 2^{128} - 159$ we calculate $\log_2 q = 7.282$ which also is in very good agreement with the observed value 7.289.

[1] Note that the formulae in Knuth on page 370 and 372 are based on the input to the Euclidean algorithm being u, m instead of m, u. This order of input means that the first quotient is always 0 and the formulae in Knuth count this as one step.

4 Conclusion

We have presented a new algorithm for rational reconstruction. Our algorithm improves Wang's algorithm by reducing the size of the modulus m needed to reconstruct a rational n/d when n and d have different length. Our algorithm is almost optimal in the sense that it requires that m be only a few bits longer than $2|n|d$. Thus we improve the efficiency of modular algorithms when we don't know the shape and size of the rational numbers we are attempting to reconstruct in advance. For modular algorithms where $m = p_1 \times p_2 \times ... \times p_k$ the improvement is a reduction of the number of primes needed by up to a factor of 2. Similarly, for modular algorithms where $m = p^k$ is a power of a prime, we reduce k by up to a factor of 2.

We have proven that the new algorithm will output n/d from its image u modulo m when $m > 9d^2n^2$ and with high probability when $m > 2^k d|n| \log_2 m$ for a suitable integer k. This latter result is based on being able to show that, over all inputs u on $[0, m)$ to the Euclidean algorithm, the length of the median maximal quotient, $\log_2 q = \log_2 \log_2 m + O(1)$. We do not have the result that we really want, however, namely a precise result that given m and k allows us to choose q such that $Pr(Q \leq q) = 2^{-k}$ so that we can set the parameter T in our algorithm.

We have implemented algorithm MQRR in Maple and used it in our Maple implementation of the modular GCD algorithm for polynomials over number fields (see van Hoeij and Monagan [5]). Our implementation sets $T = 2^{20} \log_2 m$ which works well from experiment.

Algorithm MQRR is based on the (extended) Euclidean algorithm and hence it has time complexity $O(\log^2 m)$. We expect that the efficiency improvements made to Wang's algorithm by the authors in [1] and [12] should be applicable to our algorithm, because both algorithms are based on the Euclidean algorithm.

The new algorithm can be applied to the problem of rational function reconstruction in one parameter t over a finite field GF(q) with q elements. Let $n, d \in GF(q)[t]$ be relatively prime with $d \neq 0$. Let $m \in GF(q)[t]$ be relatively prime to d. Let $u = n/d \bmod m$. We run the extended Euclidean algorithm with inputs $m, u \in GF(q)[t]$. Lemma 1 becomes $\deg r_i + \deg t_i + \deg q_{i+1} = \deg m$. The algorithm outputs the rational function r_i/t_i for Q a quotient of maximal degree (provided $\gcd(t_i, m) = 1$). If we impose that $2 \deg Q > \deg m$ then the algorithm will output n/d. If we impose that $\deg Q > 1$ then the probability it outputs n/d will be high provided q is not small; we think the probability of error is $O(q^{1-\deg Q})$.

References

[1] G. E. Collins and M. J. Encarnacion. Efficient Rational Number Reconstruction. *J. Symbolic Computation* **20**, pp. 287–297, 1995.

[2] M. J. Encarnacion. Computing GCDs of Polynomials over Algebraic Number Fields, *J. Symbolic Computation* **20**, pp. 299–313, 1995.

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

[4] The GNU Multiple Precision Arithmetic Library. Copyright, Free Software Foundation, Inc. (2002). `http://www.gnu.org/software/gmp/gmp.html`

[5] M. van Hoeij, M. Monagan. A Modular GCD Algorithm over Number Fields Presented with Multiple Field Extensions. *Proceedings of ISSAC '2002*, ACM Press, pp. 109–116, 2002.

[6] T. Jebelean. Improving the Multiprecision Euclidean Algorithm. *Proceedings of DISCO '93*, Springer-Verlag LNCS **722**, pp. 45–58, 1993.

[7] J. de Kleine, M. Monagan. A Modular Design and Implementation of Buchberger's Algorithm. *Proceedings of RWCA '02*, 2002.

[8] D. E. Knuth. *The Art of Computer Programming: Volume 2 Seminumerical Algorithms Third Edition*, Addison Wesley, section 4.5.3., 1998.

[9] L. Langemyr, S. McCallum. The Computation of Polynomial GCD's over an Algebraic Number Field, *J. Symbolic Computation* **8**, pp. 429–448, 1989.

[10] L. Langemyr. An Asymptotically Fast Probabilistic Algorithm for Computing Polynomial GCD's over an Algebraic Number Field. *Proc. of AAECC '90*, Springer-Verlag LNCS **508**, pp. 222–233, 1991.

[11] P .L Montgomery. *An FFT Extension of the Elliptic Curve Method of Factorization.* PhD thesis, University of California, Los Angeles, 1992.

[12] V .Y Pan, X. Wang. Acceleration of the Euclidean Algorithm and Extensions, *Proceedings of ISSAC '02*, ACM Press, pp. 207–213, 2002.

[13] A. Steel (2003). Private communication.

[14] P. S. Wang. A p-adic Algorithm for Univariate Partial Fractions. *Proceedings of SYMSAC '81*, ACM Press, pp 212-217, 1981.

[15] P. S. Wang, M. J. T. Guy, J. H. Davenport. *p-adic Reconstruction of Rational Numbers*. in SIGSAM Bulletin, **16**, No 2, 1982.

[16] P. S. Wang, M. J. T. Guy, J. H. Davenport. Early detection of true factors in Univariate Polynomial Factorization. *Proceedings of EUROCAL '83*, Springer-Verlag LNCS **162**, pp. 225–235.