

# Improving the Euclidean Algorithm

S.K. Lucas

Department of Mathematics and Statistics

James Madison University

Harrisonburg VA 22807

May 2007

## 1 The Algorithm

An integer  $c$  is a divisor of  $a$  if  $a = cq$  for some integer  $q$ , and is often denoted as  $c|a$ , and stated “ $c$  divides  $a$ .” Given the sets of divisors of two integers  $a$  and  $b$ , there will be some divisors in common, and of those there will be a largest, which we call the greatest common divisor, denoted  $\gcd(a, b)$ . If  $\gcd(a, b) = 1$ , we say that  $a$  and  $b$  are relatively prime.

The universally adopted method for finding greatest common divisors is the celebrated Euclidean algorithm (over a million Google hits). Assuming without loss of generality that  $a \geq b \geq 0$ , then there are unique positive integers  $q$  (the quotient) and  $r < b$  (the remainder) so that  $a = bq + r$ . It is easy to show that the common factors of  $a$  and  $b$  are also factors of  $r$ , and so  $\gcd(a, b) = \gcd(b, r)$ , where  $b \leq a$  and  $r < b$ . Eventually we reach  $\gcd(c, 0) = c$ , since every integer is a divisor of zero. A few integer divisions is all that is required, and represents the amount of work done by the algorithm.

Let us now introduce a slight change to the Euclidean algorithm, which we shall see can significantly reduce the number of steps required to find a greatest common divisor. The Euclidean algorithm relies on  $r$  having the same common divisors as  $a$  and  $b$ . But  $a - b$  will also have the same common divisors, so the improved algorithm replaces  $a$  and  $b$  by  $b$  and  $a - b$  before the integer division. We hope that less steps will be required to find the greatest common divisor – at least enough to justify an additional integer subtraction at each step.

It is perhaps surprising that such a simple change to the Euclidean algorithm is not well known. A version of the Euclidean algorithm relies on  $\gcd(a, b) = \gcd(b, a - b)$ , but

when only using subtraction the number of steps can become extremely large. Performing a subtraction before each integer division is not something discussed in the research literature, on the web, or in discrete mathematics textbooks. In fact, the only place I have seen a variant of it is in a few pages of Gazale [1], where the subtraction is performed after the division:  $\gcd(a, b) = \gcd(b, r - b)$ . However, Gazale writes the result in a more complicated form, and there is no discussion of how often the new method is an improvement, or indeed if it could even be worse. In fact, performing the subtraction before the division rather than vice versa can save an additional step over Gazale's version.

Some examples are in order. Consider finding  $\gcd(32, 18) = 2$ , first using the Euclidean algorithm,

$$\begin{aligned} \gcd(32, 18) &= \gcd(18, 14) & 32 &= 1 \cdot 18 + 14 \\ &= \gcd(14, 4) & 18 &= 1 \cdot 14 + 4 \\ &= \gcd(4, 2) & 14 &= 3 \cdot 4 + 2 \\ &= \gcd(2, 0) & 4 &= 2 \cdot 2 + 0 \\ &= 2, \end{aligned}$$

then the improved algorithm,

$$\begin{aligned} \gcd(32, 18) &= \gcd(18, 14) & 32 - 18 &= 14 & = \gcd(14, 4) & 18 &= 1 \cdot 14 + 4 \\ &= \gcd(10, 4) & 14 - 4 &= 10 & = \gcd(4, 2) & 10 &= 2 \cdot 4 + 2 \\ &= \gcd(2, 2) & 4 - 2 &= 2 & = \gcd(2, 0) & 2 &= 1 \cdot 2 + 0 \\ & & & & & & = 2. \end{aligned}$$

We have reduced the number of steps from four to three, where it looks like we have eliminated the first step from the Euclidean algorithm. Now how about  $\gcd(8559, 4224) = 3$ :

$$\begin{aligned} \gcd(8559, 4224) &= \gcd(4224, 111) & 8559 &= 2 \cdot 4224 + 111 \\ &= \gcd(111, 6) & 4224 &= 38 \cdot 111 + 6 \\ &= \gcd(6, 3) & 111 &= 18 \cdot 6 + 3 \\ &= \gcd(3, 0) & 6 &= 2 \cdot 3 + 0 \\ &= 3, \end{aligned}$$

and

$$\begin{aligned} \gcd(8559, 4224) &= \gcd(4335, 4224) & 8559 - 4224 &= 4335 & = \gcd(4224, 111) & 4335 &= 1 \cdot 4224 + 111 \\ &= \gcd(4113, 111) & 4224 - 111 &= 4113 & = \gcd(111, 6) & 4113 &= 37 \cdot 111 + 6 \\ &= \gcd(105, 6) & 111 - 6 &= 105 & = \gcd(6, 3) & 105 &= 17 \cdot 6 + 3 \\ &= \gcd(3, 3) & 6 - 3 &= 3 & = \gcd(3, 0) & 3 &= 1 \cdot 3 + 0 \\ & & & & & & = 3. \end{aligned}$$

This time the two algorithms both take four steps. Notice that the pair of numbers stays the same in both algorithms after each division, but the quotient in the improved algorithm is one less than for the Euclidean algorithm. Finally, consider  $\text{gcd}(21, 13) = 1$ :

$$\begin{aligned}
 \text{gcd}(21, 13) &= \text{gcd}(13, 8) & 21 &= 1 \cdot 13 + 8 \\
 &= \text{gcd}(8, 5) & 13 &= 1 \cdot 8 + 5 \\
 &= \text{gcd}(5, 3) & 8 &= 1 \cdot 5 + 3 \\
 &= \text{gcd}(3, 2) & 5 &= 1 \cdot 3 + 2 \\
 &= \text{gcd}(2, 1) & 3 &= 1 \cdot 2 + 1 \\
 &= \text{gcd}(1, 0) & 2 &= 2 \cdot 1 + 0 \\
 &= 1,
 \end{aligned}$$

and

$$\begin{aligned}
 \text{gcd}(21, 13) &= \text{gcd}(13, 8) & 21 - 13 &= 8 & = \text{gcd}(8, 5) & 13 &= 1 \cdot 8 + 5 \\
 &= \text{gcd}(5, 3) & 8 - 5 &= 3 & = \text{gcd}(3, 2) & 5 &= 1 \cdot 3 + 2 \\
 &= \text{gcd}(2, 1) & 3 - 2 &= 1 & = \text{gcd}(1, 0) & 2 &= 2 \cdot 1 + 0 \\
 & & & & & &= 1.
 \end{aligned}$$

This time the improved algorithm has skipped fully half of the Euclidean algorithm's steps.

Experimentation suggests that the improved algorithm usually takes fewer steps than the Euclidean algorithm, and in the remaining cases the number of steps remain the same. The reader who wishes to experiment with these algorithms may consider using the following Matlab routines, which output the number of steps taken as well as the greatest common divisor. In both routines the first lines take the absolute values of  $a$  and  $b$ , return immediately if  $a = b$ , and swap the order of terms if  $a < b$ .

```

function [a,n]=euclid(a,b)
if a<0, a=-a; end
if b<0, b=-b; end
n=0;
if (a==b), return, end
if b>a, c=a; a=b; b=c; end
while b~=0
    n=n+1; c=mod(a,b); a=b; b=c;
end

```

```

function [a,n]=euclid2(a,b)
if a<0, a=-a; end
if b<0, b=-b; end
n=0;
if (a==b), return, end
if b>a, c=a; a=b; b=c; end
while b~=0
    c=a-b;
    if c>b, a=c;
    else, a=b; b=c; end
    n=n+1; c=mod(a,b); a=b; b=c;
end

```

## 2 Why It Works

The performance of the improved algorithm is directly related to the occurrence of ones as the quotient when performing integer division. Two steps of the Euclidean algorithm applied to two numbers  $r_0 \geq r_1 \geq 0$  are

$$\begin{aligned} \gcd(r_0, r_1) &= \gcd(r_1, r_2), & r_0 &= r_1q_1 + r_2 & \text{and} \\ \gcd(r_1, r_2) &= \gcd(r_2, r_3), & r_1 &= r_2q_2 + r_3. \end{aligned}$$

Two equivalent steps of the improved algorithm (assuming  $q_1, q_2 > 1$ ) are

$$\begin{aligned} \gcd(r_0, r_1) &= \gcd(r_0 - r_1, r_1) = \gcd(r_1, r_2), & r_0 - r_1 &= r_1(q_1 - 1) + r_2 & \text{and} \\ \gcd(r_1, r_2) &= \gcd(r_1 - r_2, r_2) = \gcd(r_2, r_3), & r_1 - r_2 &= r_2(q_2 - 1) + r_3. \end{aligned}$$

As long as the quotients are greater than one, then the steps are essentially identical, as in our second example above. But if  $q_1 = 1$ , then  $r_0 - r_1 = r_2$ ,  $r_2 < r_1$ , and the improved algorithm is equivalent to

$$\gcd(r_0, r_1) = \gcd(r_2, r_1) = \gcd(r_2, r_3), \quad r_1 = r_2q_2 + r_3.$$

Compared to the Euclidean algorithm, we have skipped the step of dividing  $r_0$  by  $r_1$ . If there are two or more quotients equalling one in the Euclidean algorithm, then only every second one can be skipped in the improved algorithm. Thus the improved algorithm reduces the number of steps in the Euclidean algorithm by between zero and fifty percent depending on the quotients. The improved algorithm never uses more steps.

## 3 How Well It Works

If the improved algorithm only skipped the occasional step in the Euclidean algorithm, the extra computational effort of additional integer subtractions would mean that the improved algorithm would take longer to calculate a greatest common divisor. However, we can show that on average just under 30% of steps in the Euclidean algorithm are skipped in the improved algorithm. One way is to find  $\gcd(a, b)$  for all  $0 \leq b \leq a \leq n$  using both algorithms, and divide the difference in number of steps by the number of Euclidean steps. The results are shown in figure 3, where the  $x$ -axis is the upper bound  $n$  and the  $y$ -axis is the fraction improvement. As we can see, the improvement quickly converges to just under 30%. In fact, for  $n = 10\,000$  the improvement is 29.888%, and while the curve is increasing, it is doing so extremely slowly.

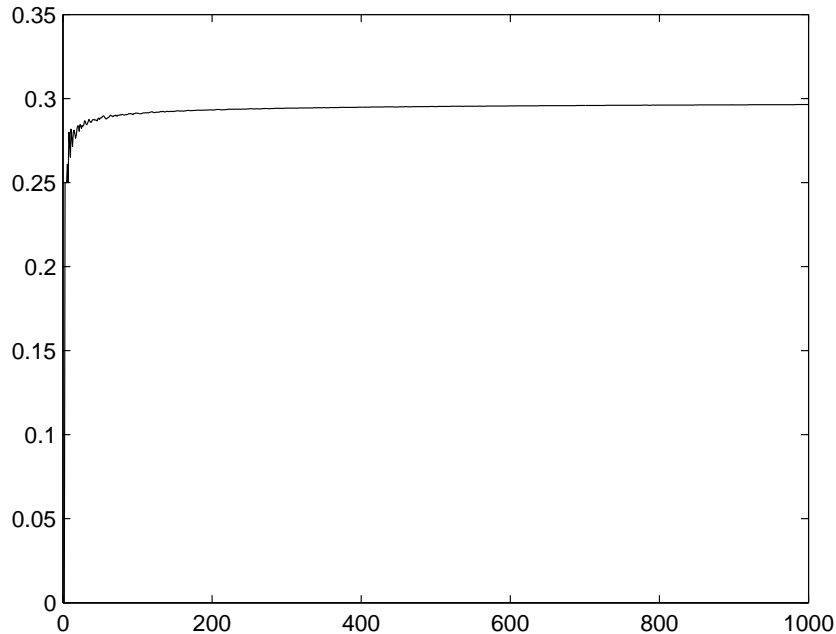


Figure 1: Fractional improvement in number of greatest common divisor steps for all integers satisfying  $0 \leq b \leq a \leq n$

We can estimate the percentage improvement theoretically. The quotients  $q_i$  obtained when applying the Euclidean algorithm are also the *partial quotients* that make up the continued fraction representation of the fraction  $x = b/a$  as

$$x = q_0 + \frac{1}{q_1 + \frac{1}{q_2 + \frac{\vdots}{q_{n-1} + \frac{1}{q_n}}}}$$

The continued fraction representation of a fraction is always finite, and the partial quotients can be found by the equivalent algorithm  $x_0 = x$  and then  $b_i = \lfloor x_i \rfloor$ ,  $x_{i+1} = 1/(x_i - b_i)$  for  $i = 0, 1, \dots, n$ , where  $\lfloor x \rfloor$  is the largest integer less than or equal to  $x$ . If  $x$  is irrational, then the iteration continues indefinitely, and we get an infinite continued fraction representation. There is a rich and fascinating structure to continued fractions, as in, for example Olds [3].

However, the feature we are particularly interested in is the Gauss-Kuzmin (or Kusmin) theorem, that states that in the continued fraction representation of almost all irrationals, the probability that any particular partial quotient takes the value  $k$  is  $\log_2 \left( 1 + \frac{1}{k(k+2)} \right)$ . For  $k = 1$  this probability is  $\log_2(4/3)$ , about 41.5037%. Gauss conjectured this result in a letter written in 1812, and it was proven correct by Kuzmin in 1928 – see for example Havil [2]. While this theorem relates to irrationals, one can argue that on average rationals display the same behavior. This is certainly born out by numerical experimentation.

In any event, it is this preponderance of ones as partial quotients, and hence quotients when employing the Euclidean algorithm, that explains the efficiency of the improved algorithm. The reason that the improvement is not over 41% is that the improved algorithm skips a step when there is a quotient one, but if the the next quotient is also one, it isn't skipped. The probability that a given quotient one is skipped is thus the probability that the previous quotient is not a one, plus the probability that the previous quotients are not a one and two ones, plus the probability that the previous quotients are not a one and four ones, etc.

Now *if* the quotients are randomly distributed and independent of each other, then the probability a given quotient equalling one is skipped can be found by summing a geometric series as  $[\log_2(4/3)(1 - \log_2(4/3))]/[1 - (\log_2(4/3))^2]$ , about 29.330%. This is slightly less than the experimental result with  $n = 10\,000$  of 29.888%, but the number of samples is sufficient to show this difference is significant. Thus, the quotients are *not* randomly distributed, which is not discussed in the literature on the Gauss-Kuzmin theorem and its extensions. It would be interesting to establish just what the correlations are between successive partial quotients in continued fractions to explain this discrepancy. However, this should not detract from the remarkable speed-up due to the improved algorithm.

## 4 When it Works Poorly

One of the well known properties of the Euclidean algorithm is that its worst case performance occurs when applied to successive Fibonacci numbers: each step of the algorithm has a quotient of one, and the numbers reduce as slowly as possible. The worst case for the improved algorithm also has quotient one, and so involves successive terms in the sequence satisfying the recurrence relations

$$P_n = 2P_{n-1} + P_{n-2} \quad \text{with} \quad P_0 = 0, P_1 = 1,$$

with explicit solution

$$P_n = \frac{(1 + \sqrt{n})^n - (1 - \sqrt{2})^n}{2\sqrt{2}}.$$

These are known as *Pell numbers* [4], and for large  $n$  the ratio of Pell to Fibonacci numbers is approaching

$$\frac{\sqrt{5}}{2\sqrt{2}} \left( \frac{2(1 + \sqrt{2})}{1 + \sqrt{5}} \right)^n \approx 0.791 \times 1.492^n.$$

Clearly, the worst case behavior of the improved algorithm is even more impressively superior to the Euclidean algorithm than the average case.

## References

- [1] M. Gazale, *Number: from Ahmes to Cantor*, Princeton University Press, Princeton NJ, 2000.
- [2] J. Havil, *Gamma: Exploring Euler's Constant*, Princeton University Press, Princeton, NJ, 2003.
- [3] C.D. Olds, *Continued Fractions*, Random House, 1963.
- [4] E.W. Weisstein, Pell Number, *from MathWorld - A Wolfram Web Resource*. <http://mathworld.wolfram.com/PellNumber.html> (February 2007).