Calculation of extended gcd by parametrization. Modular geometric sequence solving equation $au + b^n v = 1$.

WOLF Marc, WOLF François, LE COZ Corentin.

marc.wolf@tsoftemail.com
http://mathscience.tsoftemail.com

April 18, 2018

Abstract

Firstly, we propose an algorithm for calculating the extended gcd by providing a solution that minimizes one of the two coordinates. The algorithm relies on elementary arithmetic properties to parameterize the solutions.

Secondly, we propose a modular geometric sequence to solve the linear diophantine equation $au + b^n v = 1$ with a and b coprime integers and n a not null natural number.

Contents

1 Introduction

1.1 The problem of extended gcd

Let a and b be two integers, the extended gcd's problem consists in finding three integers u, v and g such that :

$$ua + vb = g$$

with g equal to gcd of a and b.

In number theory, this problem occurs in many situations like in the theory of corrector codes [BK84] or for the factorization of integers [B86]. Moreover, it is the elementary component of the classical algorithm of Smith invariants computation of a matrix with integer coefficients, which allows the general resolution of the linear diophantine equations [Ser]. See also [DSL15] for a recent application of the Euclid algorithm.

1.2 Notation

Let x and y be two integers, we denote "$x \mod y$" the remainder of the Euclidean division of x by y, which belongs to $[\![0, y-1]\!]$ by convention.

2 Extended GCD with a and b are coprime integers.

We assume therefore that a and b are *coprime* integers. Because we can reverse a and b, we will assume that a is an *odd* integer. If they are both odd, we will assume $a < b$ for reducing number of arithmetic operation.

2.1 Parameter v_c

To each integer c, we associate the following equation in u and v :

$$ua + vb = c \qquad (E_c)$$

As a reminder, the set of solutions of (E_c) is equal to $\{(u_0 + kb, v_0 - ka), k \in \mathbb{Z}\}$, with (u_0, v_0) corresponding to a particular solution.

Definition 1. we name *normal solution* of (E_c) the unique solution (u_c, v_c) with the condition $v_c \in [\![0, a-1]\!]$. We call v_c the *parameter* of (E_c).

Remark. By setting $u_c = (c - v_c b)/a$, (u_c, v_c) is the normal solution of (E_c).

From the structure of solutions of (E_c), we get the following proposition:

Proposition 1. *For any solution* (u, v) *of* (E_c), $v \equiv v_c \pmod{a}$.

2.2 Arithmetic properties of the parameter

Proposition 2. *In immediately way, the parameter is additive :*

$$\forall c, c' \in \mathbb{Z}, v_{c+c'} \equiv v_c + v_{c'} \pmod{a}$$

Corollary 1. *the parameter is multiplicative :*

$$\forall c \in \mathbb{Z}, t_c \equiv t_1 * c \pmod{a}$$

Corollary 2. *If c is even,*

$$\begin{aligned} v_{c/2} &= \frac{v_c}{2}, \textit{ if } v_c \textit{ is even} \\ &= \frac{v_c + a}{2}, \textit{ otherwise.} \end{aligned}$$

We deduce the following algorithm which will be used as an elementary component of the final algorithm: with (c, v), and v as parameter of (E_c), it returns the pair of integers obtained by dividing as many times as possible c by 2.

Algorithm 1 Function Div1(c,v)

```
while c is even do
    c ← c/2
    if v is even then
        v ← v/2
    else
        v ← (v + a)/2
    end if
end while                    ▷ we divide as many times as possible c by 2.
return (c, v)
```

The parameter is obviously *linear* in c, but we only need the following proposition :

Proposition 3.

$$\forall c, c' \in \mathbb{Z}, v_{c-c'} \equiv v_c - v_{c'} \pmod{a}$$

Proposition 4. *We determine two particular values of the parameter :*

- *for* $c = b \mod a$, $v_c = 1$
- *for* $c = -b \mod a$, $v_c = a - 1$

Proof. **Case 1 :** $c = b \mod a$
Let q be the quotient of the Euclidean division of b by a. We obtain the following equality :

$$-qa + b = c$$

ie $(-q, 1)$ is solution of (E_c). Then $v_c = 1$

Case 2 : $c = -b \mod a$

Let us proceed in a similar way. Let q' be the quotient of the Euclidean division of $-b$ by a. One gets the following equality :

$$-q'a - b = c$$

ie $(-q', -1)$ is solution of (E_c). Then $v_c = a - 1$. □

We will initialize the algorithm with these two values. A sole Euclidean division is performed before applying the binary algorithm.

2.3 Presentation of the algorithm

Solving the problem of extended gcd with a and b being coprime integers leads to determine v_1. The algorithm **WWL1** proceeds by step: starting from the two values of v_c given by the precedent proposition, it uses the corollary 2 of proposition 2 to determine v_1.

Once v_1 has been determined, the complete solution is obtained by calculating:

$$u_1 = \frac{1 - v_1 b}{a}.$$

Then (u_1, v_1) is a solution of (E_1). However, since the Euclidean division of b by a is done at the beginning of the algorithm, computational time is saved with $a < b$ by performing: $u_1 = -v_1 q + (1 - v_1 r)/a$, denoting q and r respectively the quotient and the remainder of the Euclidean division of b by a.

Algorithm 2 WWL1 Let a and b be two integers, with a and b coprimes and a an odd integer, returns a pair of integers (u,v) such that $ua+vb=1$ and $v \in [\![0, a-1]\!]$

First step : initialization.

$c_1 \leftarrow b - \lfloor \frac{b}{a} \rfloor a$, $c_2 \leftarrow a - c_1$
$v_1 \leftarrow 1$, $v_2 \leftarrow a-1$ ▷ initialization of the two values of parameter v_c.

$(c_1, v_1) \leftarrow Div1(c_1, v_1)$ ▷ we divide as many times as possible c_1 by 2.
$(c_2, v_2) \leftarrow Div1(c_2, v_2)$ ▷ we divide as many times as possible c_2 by 2.

if $c_2 < c_1$ **then**
  $(c_1, c_2) \leftarrow (c_2, c_1)$, $(v_1, v_2) \leftarrow (v_2, v_1)$
end if ▷ we ensure that $c_1 < c_2$

Second step : iteration.

while $c_1 > 1$ **do**
  $c_2 \leftarrow c_2 - c_1$
  if $v_2 - v_1 < 0$ **then**
    $v_2 \leftarrow v_2 - v_1 + a$
  else
    $v_2 \leftarrow v_2 - v_1$
  end if ▷ we modify c_2 and we compute the associated parameter v_2.

  $(c_2, v_2) \leftarrow Div1(c_2, v_2)$ ▷ we divide as many times as possible c_2 by 2.

  if $c_2 < c_1$ **then**
    $(c_1, c_2) \leftarrow (c_2, c_1)$, $(v_1, v_2) \leftarrow (v_2, v_1)$
  end if ▷ we reassign c_1 and c_2 so as to have $c_1 < c_2$.

end while ▷ the loop is left when $c_1 = 1$

$v \leftarrow v_1$
$u_1 \leftarrow -vq$, $u_2 \leftarrow (1-vr)/a$
$u \leftarrow u_1 - u_2$
return (u, v) ▷ the solution is returned

2.4 Validity of the algorithm

It can be noted that "$gcd(c_1, c_2) = gcd(a, b)$" is a loop invariant. Moreover, since each iteration of the last loop **While** c_1 or c_2 decreases, this proves the *termination* of the algorithm.

According to the precedent propositions, another invariant is: "$bv_{c_i} - c_i \equiv 0 \pmod{a}$, for $i = 1, 2$". This finally proves the *correction* of the algorithm.

3 General case

It is easy to adapt the precedent algorithm to the general case where a and b are not assumed to be coprimes. Experimental tests highlighted that it is more advantageous to calculate u at the same time as v as the algorithm works. We do not handle (v,c) with v parameter (E_c) anymore, but (u,v,c) triplets checking the conditions $ua+vb=c$ and $v \in [\![0, a-1]\!]$. We divide as many times as possible a and b by 2 (which has a low computational cost) and because we can reverse a and b, we assume that a is an *odd* integer.

Algorithm 3 Function Div2(u,v,c) : Let (u,v,c) be a triplet checking the conditions $ua+vb=c$ and $v \in [\![0, a-1]\!]$, returns a triplet (u,v,c) checking these two conditions, obtained by dividing as many times as possible c by 2. We suppose a an odd integer

---

while c is even **do**
  if v is even **then**
    $(u,v,c) \leftarrow (u/2, v/2, c/2)$
  else
    $(u,v,c) \leftarrow (\frac{u-b}{2}, \frac{v+a}{2}, \frac{c}{2})$
  end if
end while
return (u,v,c)

Proof. Let us note first that, given (u,v,c) satisfying $ua+vb=c$, $(u-b, v+a, c)$ satisfies this same equation. We suppose that c is even. Let us show that we get necessary parities to process the algorithm:

If v is even, then $ua = c - vb$ is even. As a is odd, this requires that u be even.

If v is odd :

- If u is odd, then $vb = c - ua$ is odd, and so v and b are odd integers. Then $u-b$ and $v+a$ are even integers
- If u is even, then $vb = c - ua$ is even and so b is even, and then $u-b$ and $v+a$ are even integers.

We get directly that v remains in the interval $[\![0, a-1]\!]$. □

Algorithm 4 WWL2: Let a and b be two integers, with a an odd integer, returns a triplet (u, v, g) such that $g = \text{pgcd}(a, b)$, $ua + vb = g$ and $v \in [\![0, a-1]\!]$

First step : initialization.

$c_1 \leftarrow b - \lfloor \frac{b}{a} \rfloor a$, $c_2 \leftarrow a - c_1$, $v_1 \leftarrow 1$, $v_2 \leftarrow a - 1$

$u_1 \leftarrow \frac{c_1 - v_1 b}{a}$, $u_2 \leftarrow 1 - u_1 - b$ ▷ initialization of two triplets (u, v, c)

$(u_1, v_1, c_1) \leftarrow Div2(u_1, v_1, c_1)$ ▷ we divide as many times as possible c_1 by 2.

$(u_2, v_2, c_2) \leftarrow Div2(u_2, v_2, c_2)$ ▷ we divide as many times as possible c_2 by 2.

if $c_2 < c_1$ **then**
  $(u_1, v_1, c_1, u_2, v_2, c_2) \leftarrow (u_2, v_2, c_2, u_1, v_1, c_1)$
end if ▷ we ensure that $c_1 < c_2$

Second step : iteration.

while $c_1 > 0$ **do**
  $c_2 \leftarrow c_2 - c_1$
  if $v_2 - v_1 < 0$ **then**
    $v_2 \leftarrow v_2 + a - v_1$, $u_2 \leftarrow u_2 - u_1 - b$
  else
    $v_2 \leftarrow v_2 - v_1$, $u_2 \leftarrow u_2 - u_1$
  end if
  ▷ we modify u_2 and v_2 so that they verify (E_{c_2}).

  $(c_2, v_2) \leftarrow Div2(c_2, v_2)$ ▷ we divide as many times as possible c_2 by 2.

  if $c_2 < c_1$ **then**
    $(u_1, v_1, c_1, u_2, v_2, c_2) \leftarrow (u_2, v_2, c_2, u_1, v_1, c_1)$
  end if ▷ we reassign c_1 and c_2 so as to have $c_1 < c_2$.
  $c_2 \leftarrow c_2 - c_1$

end while ▷ the loop is left when $c_1 = 0$

return (u_2, v_2, c_2) ▷ the solution is returned

Proof. c_1 and c_2 follow the same values as with the algorithm which computes the combined gcd (See Appendix), which proves that when we go out of the **While** loop, c_2 is equal to the gcd of a and b.

Moreover, "$u_i a + v_i b = c_i$ for $i = 1, 2$" is an invariant of the algorithm, which proves its correction. □

4 Solution of the equation $au + b^n v = 1$

We remind that v_c is the *parameter* of the equation (E_c) with (u_c, v_c) the normal solution of (E_c). At first, we will define the parameter t_c. Then we will prove that the solution of the Diophantine equation $au+b^n v = 1$ with a and b coprime integers and n a not null natural number, is written as a modular geometric sequence.

4.1 Parameter t_c

We let: $t_c + v_c = 0 \equiv a \pmod a$ with $t_c \in [\![0, a-1]\!]$

The pair of integers $(u_c + b, v_c - a)$ is solution of (E_c). With $x_c = u_c + b$, this solution is written as $(x_c, -t_c)$.

Definition 2. Values of t_c and v_c are defined as being complementary values modulo a.
We named *complementary normal solution* of (E_c) the unique solution $(x_c, -t_c)$ with $x_c = (c + t_c b)/a$ and checking $t_c \in [\![0, a-1]\!]$. We named t_c the *complementary parameter* of (E_c).

Remark. Because of relationship between the parameters v_c and t_c, arithmetic properties of both parameters are the same.

4.2 Modular geometric sequence

We denote $(E_1)^n$ the following equation: $au + b^n v = 1$. The complementary parameter of this equation is denoted $t_1(n)$.

Definition 3. A modular geometric sequence S_n is defined as follows:
$S_n \equiv q * S_{n-1} \pmod M$ with common ratio $q \in \mathbb{Z}^*$ and value $M \in \mathbb{N}^*$.

Proposition 5. *The complementary parameter $t_1(n)$ of $(E_1)^n$ with $n \in \mathbb{N}^*$ is written as a modular geometric sequence:*
$t_1(n) \equiv -t_1(1) * t_1(n-1)$ *(mod a)*

Proof: The complementary normal solution of equation $a(x_1(1)) - bt_1(1) = 1$ $(E_1)^1$ is $(x_1(1), -t_1(1))$. The complementary normal solution of equation $ax_1(n) - b^n t_1(n) = 1$ $(E_1)^n$ is $(x_1(n), -t_1(n))$.

We assume that there exists an integer z such that $b^{(n-1)}$ divides $t_1(1) + za$.

We let: $t_1(n) = (t_1(1) + za)/b^{(n-1)}$ with $t_1(n) \in [\![0, a-1]\!]$. The complementary normal solution of $(E_1)^n$ is then $(x_1(1) + bz, -t_1(n))$.

The precedent $t_1(n)$ relation leads to solving the equation $az + b^{(n-1)}(-t_1(n)) = -t_1(1)$ denoted $(E_{-t_1(1)})^{(n-1)}$

The a and $b^{(n-1)}$ coefficients are coprimes. Then there exists a pair of integers $(z, -t_1(n))$ that solves the equation. Value $t_1(n)$ can be written as follows: $t_1(n) = t_{-t_1(1)}(n-1)$. Parameter t_c is multiplicative (Corollary 1), so we get: $t_1(n) \equiv -t_1(1) * t_1(n-1) \pmod{a}$.

The parameter $t_1(n)$ follows a goemetric sequence modulo a with first term equal to $t_1(1)$ and common ratio equal to $-t_1(1)$. So $t_1(n)$ is written :

$t_1(n) \equiv (-1)^{(n-1)} * (t_1(1))^n \pmod{a}$

Corollary 3. *the parameter $t_1(n) \in [\![0, a-1]\!]$ is written depending on parity of n:*
***Case 1**: $\forall k \in \mathbb{N}^*$, $n = 2k$: $t_1(n) \equiv a - ((t_1(1))^{2k}$ (mod a))*
***Case 2**: $\forall k \in \mathbb{N}$, $n = 2k+1$: $t_1(n) \equiv (t_1(1))^{2k+1}$ (mod a)*

Corollary 4. *the parameter $v_1(n)$ of $(E_1)^n$ is written as modular geometric sequence: $v_1(n) \equiv v_1(1) * v_1(n-1)$ (mod a) and so $v_1(n) \equiv (v_1(1))^n$ (mod a).*

5 Conclusion

The solutions of a linear Diophantine equation depend on one parameter denoted v_c such that: $(c - v_c b + hb, v_c - ha)$ with $h \in \mathbb{Z}$ and $v_c \in [\![0, a-1]\!]$. Our algorithm, based on the determination of this parameter, controls the size of the final result but also the size of all the computational intermediaries.

The complementary parameter of a linear Diophantine equation $au + bv = 1$ with a and b coprime integers is defined as $t_1 + v_1 \equiv a \pmod{a}$. When the b integer is raised to the power n, $n \in \mathbb{N}^*$, the parameter $t_1(n)$ is written as a modular geometric sequence.

6 References

[B86] R. P. Brent. An improved Monte Carlo factorization algorithm. BIT 20, 176-184 (1980).

[BK84] R. P. Brent, H. T. Kung, Systolic VLSI arrays for polynomial GCD computations. IEEE Trans. Comput. C-33, 731-736 (1984)

[DLS15] Doran, Lu, Smith. New algorithms for modular inversion and representation by binary quadrtic forms arising from structure in the Euclidean algorithm, 2015. arXiv:1408.4638v2

[Ser] Denis Serre. Matrices, theory and Applications. Springer 2010.

[L16] H. Leung, A Note on Extended Euclid's Algorithm, arXiv:1607.00106, 2016

[B99] R. P. Brent. Further analysis of the Binary Euclidean algorithm. PRG TR-7-99. 1999

7 Appendix : Euclid algorithms

According to our knowledge the two main algorithms for the calculation of gcd are the Euclidean algorithm [L16] and its binary version [B99]. A combined algorithm which performs Euclidean divisions and divisions by 2 is presented.

Algorithm 5 ***Binary*** **Euclid algorithm: BinaryGCD**

Let a **et** b **be two integers, return the gcd**

$m = 0$
while a and b are even **do**
  $m \leftarrow m+1, a, b \leftarrow a/2, b/2$
end while ▹ Computation of common power of 2

$r_1, r_2 \leftarrow \min(a,b), \max(a,b)$
while $r_1 > 0$ **do**
  $r_2 \leftarrow r_2 - r_1$
  while r_2 is even **do**
    $r_2 \leftarrow r_2/2$
  end while
  $r_1, r_2 \leftarrow \min(r_1, r_2), \max(r_1, r_2)$
end while

return $r_2 \times 2^m$

Algorithm 6 ***Combined*** **Euclid algorithm**

Let a **and** b **be two integers, return the gcd**

$m = 0$
while a and b are even **do**
  $m \leftarrow m+1, a, b \leftarrow a/2, b/2$
end while ▹ Computation of the common power of 2

$r_1, r_2 \leftarrow \min(a,b), \max(a,b)$
$(r1, r2) \leftarrow (r2 \mod r1, r1)$ ▹ We perform one Euclidean division
return $BinaryGCD(r_1, r_2)$ ▹ We apply the binary algorithm