SCIENTIFIC PUBLICATIONS OF THE STATE UNIVERSITY OF NOVI PAZAR
SER. A: APPL. MATH. INFORM. AND MECH. vol. 4, 2 (2012), 41-44.

Modification of Euclidian Algorithm for Solving Modular Equations

I. Ž. Milovanović, Ć. B. Dolićanin, M. K. Stojčev, E. I. Milovanović

Abstract: In this paper we propose a modification of extended Euclid's algorithms with aim to reduce the number of iteration steps when solving modular equations. Obtained result is used to solve the system of linear modular equations in one variable (Chinese Remainder Theorem). The proposed modification is convenient for parallel implementation.

Keywords: Euclidian algorithm, residue, Chinese reminder theorem.

1 Introduction

The Euclidean algorithm (also called Euclid's algorithm) is an efficient method for computing the greatest common divisor (GCD) of two integers a and b. It is one of the oldest numerical algorithms still in common use [1, 2]. The algorithm has many theoretical and practical applications. It is a key element of the RSA algorithm, a public-key encryption method widely used in electronic commerce. It is used to solve Diophantine equations, such as finding numbers that satisfy multiple congruences (Chinese remainder theorem) or multiplicative inverses of a finite field. It can also be used to construct continued fractions, in the Sturm chain method for finding real roots of a polynomial, and in several modern integer factorization algorithms. Finally, it is a basic tool for proving theorems in modern number theory, such as Lagrange's four-square theorem and the fundamental theorem of arithmetic (unique factorization) [3, 4].

In this paper we propose one modification of extended Euclidian algorithm for solving modular equations and system of modular equations. The modification is based on the usage of residue or residue complement, where appropriate, which can reduce the number of iteration steps substantially.

Manuscript received March 17, 2012; accepted May 25,2012.

I. Ž. Milovanović, M. K. Stojčev and E. I. Milovanović are with the Faculty of Electronic Engineering, University of Niš, Niš, Serbia; Ć. B. Dolićanin is with the State University of Novi Pazar, Novi Pazar, Serbia.

2 Solving of modular equations

Let us consider a modular equation in one variable

$$ax \equiv c(\bmod b) \tag{1}$$

where a, b and c are integers so that $\mathrm{GCD}(a,b)$ divides c, denoted as $\mathrm{GCD}(a,b)|c$. Suppose X_0 is an arbitrary particular solution of equation(1). Then general solution is of the form

$$x = \frac{C}{\mathrm{GCD}(a,b)}X_0 + \frac{b \cdot t}{\mathrm{GCD}(a,b)}, \qquad t \in Z.$$

This means that to solve the equation (1) it is necessary to know X_0 and $\mathrm{GCD}(a,b)$. These are usually determined by the extended Euclidian algorithm (see [2, 3, 5, 6]).

Algorithm_1 (Extended Euclidian)

$r1 := a;; \quad r2 := b;$
$x1 := 1; \quad x2;= 0;$ } initialization

while $(r \neq 0)$ **do**
{
$q := \lfloor \frac{r_1}{r_2} \rfloor$**;**
$r := r1 - q * r2$**;**
$r1 := r2; \quad r2 := r$**;**
$x := x1 - q * x2$**;**
$x1 := x2; \quad x2 := x$**;**
}
$GCD := r1; \quad X_0 := x1;$

In order to reduce the number of iteration steps, we propose the following modification of Algorithm_1

Algorithm_2 (Modified extended Euclidian)

$r1 := a;;$ $r2 := b;$
$x1 := 1;$ $x2;= 0;$
$rr := a \bmod b;$ $rs := b - rr;$ } initialization
/* rr is residuo; rs is residuo complement */

while $(rr \wedge rs \neq 0)$ **do**
if $(rr < rs)$ **then**
{ $q := \lfloor \frac{r_1}{r_2} \rfloor$**;**
$rr := r1 - q * r2$**;**
$r1 := r2; \quad r2 := rr$**;**

$x := x1 - q * x2$**;**
$x1 := x2; \quad x2 := x$**;** $\}$
else
$\{\ q := \lceil \frac{r_1}{r_2} \rceil$**;**
$rs := r1 - q * r2$**;**
$r1 := r2; \quad r2 := rs$**;**
$x := q * x2 - x1$**;**
$x1 := x2; \quad x2 := x$**;** $\}$
$GCD := r1; \quad X0 := x1;$

For the sake of illustration of running Algorithm_1 and 2, we will take the following equation

$$233x \equiv 7 (\bmod 144) \tag{2}$$

Table 1 outlines execution steps when Algorithm_1 and Algorithm_2 are used.

Table 1.

Algorithm_1								Algorithm_2						
step	q	r_1	r_2	r	x_1	x_2	x	q	r_1	r_2	r	x_1	x_2	x
1	1	233	144	89	1	0	1	2	233	144	55	1	0	-1
2	1	144	89	55	0	1	-1	3	144	55	21	0	-1	-3
3	1	89	55	34	1	-1	2	3	55	21	8	-1	-3	-8
4	1	55	34	21	-1	2	-3	3	21	8	3	-3	-8	-21
5	1	34	21	13	2	-3	5	3	8	3	1	-8	-21	-55
6	1	21	13	8	-3	5	-8	3	3	1	0	-21	-55	-309
7	1	13	8	5	5	-8	13		1		-55		89	
8	1	8	5	3	-8	13	-21	$GDC(233,144) = 1 \qquad X_0 = -55$						
9	1	5	3	2	13	-21	34							
10	1	3	2	1	-21	34	-55							
11	2	2	1	0	34	-55	149							
12		1	0		-55		89							
$GDC(233,144) = 1 \qquad X_0 = -55$														

As can be seen from Table 1, Algorithm_1 requires 12 computational steps, while Algorithm_2 requires 7 steps. Algorithm_2 uses one additional testing at the beginning of the loop. All other computational steps in the loop body of both algorithms are of identical complexity. This obviously justifies the usage of the involved modification.

In the sequel we will show how Algorithm_2 can be used for solving the system of modular equations in one variable based on Chinese remainder theorem.

Consider the following system of modular equations

$$\begin{aligned} x &\equiv c_1 (\bmod b_1) \\ x &\equiv c_2 (\bmod b_2) \end{aligned} \tag{3}$$

$$\begin{aligned} &\vdots \\ x \quad &\equiv \quad c_k(\bmod b_k) \end{aligned}$$

where b_i and c_i are integers, and b_i, $i = 1,2,\ldots,n$ are pairwise relatively prime. If we denote by x_j, $j = 1,2,\ldots,n$, solutions of the corresponding modular equations in system (3), i.e.

$$a_j x_j \equiv c_j(\bmod b_j), \qquad a_j = \frac{b_1 b_2 \ldots b_k}{b_j} = \frac{b}{b_j}, \tag{4}$$

then a particular solution of system (3) is given by

$$x \equiv \left(\sum_{j=1}^{k} a_j x_j \right) \bmod b. \tag{5}$$

If minimal positive solution of system (3) is required, then in (5) instead of mod a function Mod defined in [7] should be used.

Finally, let us note that computations defined by (4) have high degree of data parallelism and can be performed in parallel since the computations of x_j, $j = 1,2,\ldots,n$ are completely independent from each other (see for example [8, 9].

References

[1] J. M. ANDERSON, *Discrete mathematics with combinatorics*, Prentice Hall, New Jersey, 2004.

[2] D. KNUTH, *The art of computer programming*, Vol. 2, Semi-numerical algorithms, Reading MA: Addison-Wesley, 1981.

[3] A. G. AKRITAS, *Elements of computer algebra with applications*, John Wiley and Sons, Inc., New York, 1989.

[4] S. M. KUO, B. H. LEE, W. TIAN, *Real-time digital signal processing: Implementations and applications*, John Wiley and Sons, Inc. 2006.

[5] R. P. BRENT, H. T. KUNG, *Systolic VLSI arrays for polynomial GCD computation*, Report CMU-CS-82-118, Carnegie-Mellon University, 1982.

[6] R. P. BRENT, H. T. KUNG, F. T. LUK, *Some linear-time algorithms for systolic arrays*, arXiv: 1004.3716V1,[CS.DS], 21. Apr. 2010.

[7] M. K. STOJČEV, E. I. MILOVANOVIĆ, I. Ž. MILOVANOVIĆ, *A unified approach in manipulation with modular arithmetic*, Proc.: 28 International Conference on Microelectronics, (MIEL'12), Niš, Serbia, 2012, 387-392.

[8] S. G. AKL, *The design and analysis of parallel algorithms*, Prentice-Hall Inc. 1989.

[9] A. BORODIN, J. GATHEN, J. HOPCROFT, *Fast parallel matrix and GCD computations*, Proc. : 23rd Annual Symposium on Foundations of Computer Science, IEEE, New York, 1982, 65-71.