MATHEMATICS OF COMPUTATION
VOLUME 44, NUMBER 170
APRIL 1985, PAGES 519–521

Modular Multiplication Without Trial Division

By Peter L. Montgomery

Abstract. Let $N > 1$. We present a method for multiplying two integers (called *N-residues*) modulo N while avoiding division by N. N-residues are represented in a nonstandard way, so this method is useful only if several computations are done modulo one N. The addition and subtraction algorithms are unchanged.

1. Description. Some algorithms [1], [2], [4], [5] require extensive modular arithmetic. We propose a representation of residue classes so as to speed modular multiplication without affecting the modular addition and subtraction algorithms.

Other recent algorithms for modular arithmetic appear in [3], [6].

Fix $N > 1$. Define an *N-residue* to be a residue class modulo N. Select a radix R coprime to N (possibly the machine word size or a power thereof) such that $R > N$ and such that computations modulo R are inexpensive to process. Let R^{-1} and N' be integers satisfying $0 < R^{-1} < N$ and $0 < N' < R$ and $RR^{-1} - NN' = 1$.

For $0 \leqslant i < N$, let i represent the residue class containing $iR^{-1} \bmod N$. This is a complete residue system. The rationale behind this selection is our ability to quickly compute $TR^{-1} \bmod N$ from T if $0 \leqslant T < RN$, as shown in Algorithm REDC:

function REDC(T)
  $m \leftarrow (T \bmod R)N' \bmod R$ [so $0 \leqslant m < R$]
  $t \leftarrow (T + mN)/R$
  if $t \geqslant N$ **then return** $t - N$ **else return** t ∎

To validate REDC, observe $mN \equiv TN'N \equiv -T \bmod R$, so t is an integer. Also, $tR \equiv T \bmod N$ so $t \equiv TR^{-1} \bmod N$. Thirdly, $0 \leqslant T + mN < RN + RN$, so $0 \leqslant t < 2N$.

If R and N are large, then $T + mN$ may exceed the largest double-precision value. One can circumvent this by adjusting m so $-R < m \leqslant 0$.

Given two numbers x and y between 0 and $N - 1$ inclusive, let $z = \text{REDC}(xy)$. Then $z \equiv (xy)R^{-1} \bmod N$, so $(xR^{-1})(yR^{-1}) \equiv zR^{-1} \bmod N$. Also, $0 \leqslant z < N$, so z is the product of x and y in this representation.

Other algorithms for operating on N-residues in this representation can be derived from the algorithms normally used. The addition algorithm is unchanged, since $xR^{-1} + yR^{-1} \equiv zR^{-1} \bmod N$ if and only if $x + y \equiv z \bmod N$. Also unchanged are

Received December 19, 1983.
1980 *Mathematics Subject Classification*. Primary 10A30; Secondary 68C05.
Key words and phrases. Modular arithmetic, multiplication.

the algorithms for subtraction, negation, equality/inequality test, multiplication by an integer, and greatest common divisor with N.

To convert an integer x to an N-residue, compute $xR \bmod N$. Equivalently, compute REDC$((x \bmod N)(R^2 \bmod N))$. Constants and inputs should be converted once, at the start of an algorithm. To convert an N-residue to an integer, pad it with leading zeros and apply Algorithm REDC (thereby multiplying it by $R^{-1} \bmod N$).

To invert an N-residue, observe $(xR^{-1})^{-1} \equiv zR^{-1} \bmod N$ if and only if $z \equiv R^2x^{-1} \bmod N$. For modular division, observe $(xR^{-1})(yR^{-1})^{-1} \equiv zR^{-1} \bmod N$ if and only if $z \equiv x(\text{REDC}(y))^{-1} \bmod N$.

The Jacobi symbol algorithm needs an extra negation if $(R/N) = -1$, since $(xR^{-1}/N) = (x/N)(R/N)$.

Let $M|N$. A change of modulus from N (using $R = R(N)$) to M (using $R = R(M)$) proceeds normally if $R(M) = R(N)$. If $R(M) \neq R(N)$, multiply each N-residue by $(R(N)/R(M))^{-1} \bmod M$ during the conversion.

2. Multiprecision Case. If N and R are multiprecision, then the computations of m and mN within REDC involve multiprecision arithmetic. Let b be the base used for multiprecision arithmetic, and assume $R = b^n$, where $n > 0$. Let $T = (T_{2n-1}T_{2n-2} \cdots T_0)_b$ satisfy $0 \leqslant T < RN$. We can compute $TR^{-1} \bmod N$ with n single-precision multiplications modulo R, n multiplications of single-precision integers by N, and some additions:

```
c ← 0
for i := 0 step 1 to n − 1 do
    (dT_{i+n−1} ··· T_i)_b ← (0T_{i+n−1} ··· T_i)_b + N*(T_i N′ mod R)
    (cT_{i+n})_b ← c + d + T_{i+n}
    [T is a multiple of b^{i+1}]
    [T + cb^{i+n+1} is congruent mod N to the original T]
    [0 ≤ T < (R + b^i)N]
end for
if (cT_{2n−1} ··· T_n)_b ≥ N then
    return (cT_{2n−1} ··· T_n)_b − N
else
    return (T_{2n−1} ··· T_n)_b
end if
```

Here variable c represents a delayed carry—it will always be 0 or 1.

3. Hardware Implementation. This algorithm is suitable for hardware or software. A hardware implementation can use a variation of these ideas to overlap the multiplication and reduction phases. Suppose $R = 2^n$ and N is odd. Let $x = (x_{n-1}x_{n-2} \cdots x_0)_2$, where each x_i is 0 or 1. Let $0 \leqslant y < N$. To compute $xyR^{-1} \bmod N$, set $S_0 = 0$ and S_{i+1} to $(S_i + x_iy)/2$ or $(S_i + x_iy + N)/2$, whichever is an integer, for $i = 0, 1, 2, \ldots, n-1$. By induction, $2^iS_i \equiv (x_{i-1} \cdots x_0)y \bmod N$ and $0 \leqslant S_i < N + y < 2N$. Therefore $xyR^{-1} \bmod N$ is either S_n or $S_n - N$.

System Development Corporation
2500 Colorado Avenue
Santa Monica, California 90406

1. J. M. POLLARD, "Theorems on factorization and primality testing," *Proc. Cambridge Philos. Soc.*, v. 76, 1974, pp. 521–528.

2. J. M. POLLARD, "A Monte Carlo method for factorization," *BIT*, v. 15, 1975, p. 331–334.

3. GEORGE B. PURDY, "A carry-free algorithm for finding the greatest common divisor of two integers," *Comput. Math. Appl.* v. 9, 1983, pp. 311–316.

4. R. L. RIVEST, A. SHAMIR & L. ADLEMAN, "A method for obtaining digital signatures and public-key cryptosystems," *Comm. ACM*, v. 21, 1978, pp. 120–126; reprinted in *Comm. ACM*, v. 26, 1983, pp. 96–99.

5. J. T. SCHWARTZ, "Fast probabilistic algorithms for verification of polynomial identities," *J. Assoc. Comput. Mach.*, v. 27, 1980, pp. 701–717.

6. GUSTAVUS J. SIMMONS, "A redundant number system that speeds up modular arithmetic," Abstract 801-10-427, *Abstracts Amer. Math. Soc.*, v. 4, 1983, p. 27.