Avoiding adjustments in modular computations

Paul Zimmermann

March 2, 2012

We consider a sequence of operations (additions, subtractions, multiplications) modulo a fixed integer N, where only the final value is needed, therefore intermediate computations might use any representation. This kind of computation appears for example in number theoretic transforms (NTT) [2], in stage 1 of the elliptic curve method for integer factorization [3], in modular exponentiation, ... Our aim is to avoid as much as possible *adjustment steps*, which consist in adding or subtracting N, since those steps are useless in the mathematical sense.

Assuming residues modulo N are uniformly random, we measure the average number of adjustment steps for each addition (or subtraction) and each multiplication.

We assume the number N is stored on n machine words, and residues modulo N are stored in a sign-magnitude representation, as in GMP [1]. We denote by β the smallest power of the word base which is larger than N.

We assume multiplications use Montgomery's reduction $ab\beta^{-1} \bmod N$, where $N < \beta$. We will denote $\varepsilon = N/\beta$.

We will compare the different methods theoretically, and confirm experimentally the obtained figures with the two numbers $N_1 = 4670326759$ and $N_2 = 7675265546198221715$ in base $\beta = 2^{32}$, with respectively $\varepsilon_1 \approx 2.5 \cdot 10^{-10}$ and $\varepsilon_2 \approx 0.42$. Our test program is the following: it computes $a = 2^{F_{k+1}} \bmod N$ and $b = 2^{F_k} \bmod N$, where F_k is the k-th Fibonacci number, $F_0 = 0$, $F_1 = 1$, $F_k = F_{k-1} + F_{k-2}$ for $k \geq 2$:

```
a = 2; b = 1; c = 1
for k := 1 to 10^6
   (a, b) = (mulmod(a,b,N), a)
   if odd(k) then c = addmod(a,b,N) else c = submod(a,b,N)
```

For N_1 at the end of this test program we should obtain $a = 4241733463$, $b = 4461431479$, $c = 4450628743$; for N_2 at the end we should obtain $a = 6410185500671098032$, $b = 5369541078340869818$, $c = 1040644422330228214$.

1 Non-negative non-redundant representation

Each residue is in the interval $[0, N-1]$.

For an addition $a+b$, the probability that $a+b \geq N$ is $1/2$, thus we get an adjustment with probability $1/2$:

```
c = a + b
if c >= N:
   c = c - N
```

For a subtraction $a-b$, we have $a < b$ with probability $1/2$ too:

```
c = a - b
if c < 0:
   c = c + N
```

For a modular multiplication, we assume we have precomputed $m = -1/N \bmod \beta$:

```
c = a * b
q = m * c mod B
e = (c + qN) / B           % exact division
if e >= N:
   e = e - N
```

We have $0 \leq q < \beta$, thus $0 \leq c + qN < N^2 + \beta N$, and $0 \leq (c+qN)/\beta < N^2/\beta + N$.

If $N = \varepsilon\beta$, then $(c+qN)/\beta < (1+\varepsilon)N$. Assuming $a = xN$, $b = yN$, and $q = z\beta$, we get: $(c+qN)/\beta = (xy\varepsilon + z)N$, thus the probability of adjustment is that of $xy\varepsilon + z \geq 1$, i.e., $1 - xy\varepsilon \leq z < 1$, which is $\varepsilon/4$:

```
sage: var('x y e'); integrate(integrate(x*y*e,(x,0,1)),(y,0,1))
1/4*e
```

2 Symmetric non-redundant representation

Each residue is in the interval $-N/2 \leq a < N/2$, which contains exactly N consecutive integer values, N being even or odd. The addition is as follows:

```
c = a + b
if c >= N/2:
   c = c - N
else if c < -N/2:
   c = c + N
```

Let $a = xN$ and $b = yN$ with $-1/2 \le x, y \le 1/2$, an adjustment occurs when $|x + y| \ge 1/2$.

```
sage: var('x y'); 2*integrate(integrate(1,(y,1/2-x,1/2)),(x,0,1/2))
1/4
```

(The factor 2 takes into account the case $x + y < -1/2$.) For a subtraction it is similar.

For a modular multiplication, we use the following algorithm, with $m = -1/N \bmod \beta$, and where `q = m * c mods B` means that $-\beta/2 \le q < \beta/2$:

```
c = a * b
q = m * c mods B
e = (c + qN) / B
if e >= N/2:
   e = e - N
elif e < -N/2:
   e = e + N
return e
```

Now c is in $[-N^2/4, N^2/4]$, q is in $[-\beta/2, \beta/2[$, qN is in $[-\beta N/2, \beta N/2[$, thus $c + qN$ is in $[-N^2/4 - \beta N/2, N^2/4 + \beta N/2]$, and $(c + qN)/\beta$ is in $[(-1/2 - \varepsilon/4)N, (1/2 + \varepsilon/4)N]$. We have $(c + qN)/\beta = (xy\varepsilon + z)N$, with $-1/2 \le x, y, z \le 1/2$, and an adjustment is needed when $|xy\varepsilon + z| \ge 1/2$, i.e., $xy \ge 0$ and $z \ge 1/2 - xy\varepsilon$ or $xy \le 0$ and $z \le -1/2 - xy\varepsilon$:

```
sage: var('x y e'); 4*integrate(integrate(x*y*e,(x,0,1/2)),(y,0,1/2))
1/16*e
```

The adjustment probability for a multiplication is thus $\varepsilon/16$, where the factor 4 takes into account $-1/2 \le x, y \le 0$, and the case where x and y are of opposite signs.

Note however that if we use a sign-magnitude representation, the computation of `q = m * c mods B` will first compute $q' = mc \bmod \beta$ with $0 \le q' < \beta$, and then subtract β if $q' \ge \beta/2$. This is some kind of adjustment we should avoid, or take into account.

3 Non-negative word-aligned redundant representation

Since most low-level operations in GMP have a cost which only depends of the number of words[1] of the operands, we can allow the residues to be

[1] called *limbs* in GMP

redundant, as long as they use the same number of words of the modulus N. In other words we allow $0 \leq a < \beta$.

The addition works as follows, with $kN < \beta \leq (k+1)N$:

```
c = a + b
if c >= B:
   c = c - kN
   if c >= B
      c = c - N
```

Assuming a, b are uniformly distributed in $[0, \beta-1]$, the probability of adjustment by kN is $1/2$. The second adjustment happens when $a + b \geq \beta + kN$, which implies $a + b \geq 2\beta - N = (2 - \varepsilon)\beta$, which occurs with probability $\varepsilon^2/2$.

The subtraction is similar:

```
c = a - b
if c < 0:
   c = c + kN
   if c < 0:
      c = c + N
```

For the multiplication, the algorithm is almost the same as for the non-negative non-redundant representation:

```
c = a * b
q = m * c mod B
e = (c + qN) / B
if e >= B:
   e = e - N
```

We have $0 \leq q < \beta$, thus $0 \leq c+qN < \beta^2+\beta N$, and $0 \leq (c+qN)/\beta < \beta+N$.

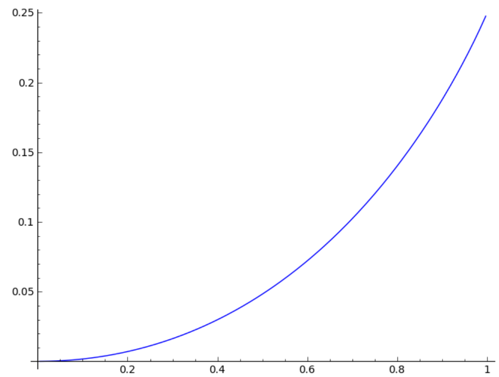
If $a = x\beta$, $b = y\beta$, and $q = z\beta$, an adjustment occurs when $xy + \varepsilon z \geq 1$, i.e., $(1 - xy)/\varepsilon \leq z \leq 1$.

```
sage: var('x y e'); assume(e-1<0);
sage: integrate(integrate(1-(1-x*y)/e,(x,(1-e)/y,1)),(y,1-e,1)).expand()
-1/2*e*log(-e + 1) + 3/4*e - 1/2*log(-e + 1)/e + log(-e + 1) - 1/2
```

We thus get for the adjustment probability for a multiplication, assuming uniform inputs in $[0, \beta - 1]$:

$$\frac{3}{4}\varepsilon - \frac{1}{2} + (1 - \varepsilon/2 - 1/(2\varepsilon)) \log(1 - \varepsilon). \tag{1}$$

This probability goes from 0 for $\varepsilon = 0$ to $1/4$ for $\varepsilon = 1$:



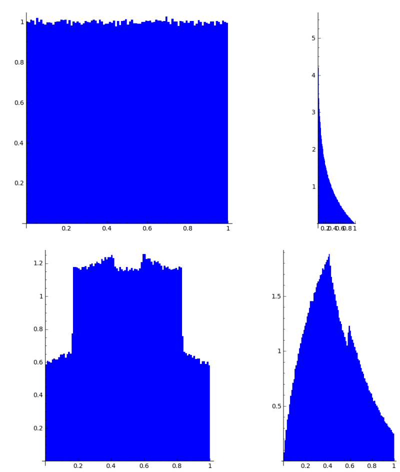


Figure 1: Distribution of the output for the addition (left) and the multiplication (right) for uniformly distributed input for N_1 (up) and N_2 (down) for the non-negative word-aligned representation.

Note however that the above assumes that the inputs of each operation are uniformly distributed. This is no longer true for a sequence of operations. For small ε, for inputs uniformly distributed in $[0, \beta - 1]$, the output of the addition or subtraction is almost uniform in $[0, \beta - 1]$, since the second adjustment is rare (see Fig. 1, up left); however when an adjustment occurs in the multiplication, the output is then in $[0, N - 1]$ which is much smaller, thus the input of the following addition or subtraction is not uniform in $[0, \beta - 1]$ (see Fig. 1, up right). Similarly, for "large" ε, for inputs uniformly distributed in $[0, \beta-1]$, the output of the addition or subtraction is no longer uniform in $[0, \beta - 1]$ (see Fig. 1, down left).

Therefore the overall adjustment probability depends on the actual sequence of operations, in particular on the ratio of additions vs multiplications.

4 Symmetric word-aligned redundant representation

Here we allow $-\beta < a < \beta$ (we could restrict to $-\beta/2 \le a < \beta/2$, but it is simpler to compare the absolute value of a to β).

The addition works as follows, with $kN < \beta \le (k+1)N$:

```
c = a + b
if c >= B:
   c = c - kN
   if c >= B
      c = c - N
elif c <= -B:
   c = c + kN
   if c <= -B:
      c = c + N
```

Assuming a, b are uniformly distributed in $[1-\beta, \beta-1]$, the probability of adjustment is $1/4$. The subtraction is exactly the same, with the first line changed into `c = a - b`.

For the multiplication, the algorithm is the following, with $m = 1/N \bmod \beta$:

```
c = |a| * |b|
q = m * c mod B
e = (c - qN) / B
return sign(a)*sign(b)*e
```

We have $0 \le c < \beta^2$, $0 \le q < \beta$, thus $-\beta N < c - qN < \beta^2$, and $-N < (c - qN)/\beta < \beta$. Therefore no adjustment is needed for e.

When each result of an addition or subtraction is given as input of a multiplication (and never to another addition or subtraction), in some cases the residues will never reach β in absolute value. This is because Montgomery reduction is contracting. Assume $|a|, |b| < \alpha N$ before an addition (or subtraction), for some $\alpha \ge 1$, then the result of the addition is

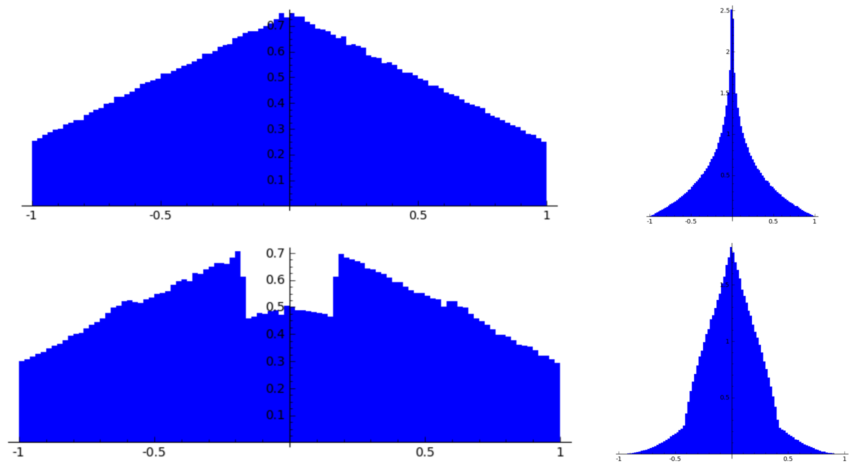


Figure 2: Distribution of the output for the addition (left) and the multiplication (right) for uniformly distributed input for N_1 (up) and N_2 (down) for the symmetric word-aligned representation.

bounded by $2\alpha N$. Thus in the following multiplication $0 \le c < 4\alpha^2 N^2$, and $-\beta N < c - qN < 4\alpha^2 N^2$, and $-N < (c - qN)/\beta < 4\alpha^2 \varepsilon N$. Thus for $\varepsilon < 1/4$, and $\alpha = 1/(4\varepsilon)$, the inputs of an addition are bounded — in absolute value — by $\alpha N = \beta/4$, thus the addition result is bounded by $\beta/2$; and the outputs of a multiplication are bounded by $4\alpha^2 \varepsilon N = \alpha N = \beta/4$ again. Thus no reduction at all is needed.

Going back to the non-negative word-aligned redundant representation, we have a similar phenomenon for the additions, with $\alpha = 2$ for $\varepsilon < 1/16$, thus no adjustment occurs in that case in the additions. However for the *subtractions* this is not the case, since with probability $1/2$ the value of $a - b$ is negative, and one adjustment is needed to get a non-negative residue. A symmetric representation avoids this.

5 Summary

The following table gives theoretical probabilities of adjustment for additions/subtractions and multiplications for each of the four representations, assuming the inputs of each operation are uniformly distributed in the allowed range.

representation	add/sub	mul
non-negative non-redundant	$1/2$	$\varepsilon/4$
symmetric non-redundant	$1/4$	$\varepsilon/16$
non-negative word-aligned	$1/2+\varepsilon^2/2$	Eq. (1)
symmetric word-aligned	$1/4$	0

The last table gives experimental results with the test program given above, and the two test numbers N_1 and N_2. The results for the non-negative non-redundant representation match exactly the theory, with $\varepsilon_1/4 \approx 6 \cdot 10^{-11}$ and $\varepsilon_2/4 \approx 0.104$. Similarly for the symmetric non-redundant representation, with $\varepsilon_1/16 \approx 1.6 \cdot 10^{-11}$ and $\varepsilon_2/16 \approx 0.026$. For the non-negative word-aligned representation, we get an average of about $1/4$ adjustment for the additions and subtractions for N_1 since no adjustment is done for the additions because $N < \beta/16$, and half of the subtractions yield an adjustment; for N_2 the 0.317896 figure includes 0.250448 for the subtractions, for the same reason as above, and we have extra adjustments in the additions and multiplications, since $\varepsilon_2 > 1/16$. Finally for the symmetric word-aligned representation we have no adjustment at all for N_1 since $\varepsilon_1 < 1/4$.

The fact that we do not get any adjustment for N_2 despite $\varepsilon_2 > 1/4$ is due to the special form of our test program. Indeed, we only perform multiplications on a and b, in which case it follows that $|a|, |b| < N$; then since $N_2 < \beta/2$ it easily follows that $a+b$ cannot exceed β in absolute value.

	N_1		N_2	
representation	add/sub	mul	add/sub	mul
non-negative non-redundant	0.500060	0	0.500182	0.104491
symmetric non-redundant	0.249616	0	0.249211	0.026052
non-negative word-aligned	0.249892	0	0.317896	0.006510
symmetric word-aligned	0	0	0	0

Acknowledgements. Many thanks to David Harvey and Paul Leyland for pointers and discussions concerning this topic.

References

[1] *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.4 ed., 2011. `http://gmplib.org/`.

[2] Harvey, D. Faster arithmetic for number-theoretic transforms. slides presented at the FLINT/Sage Days 35, Warwick, 2011.

[3] Lenstra, H. W. Factoring integers with elliptic curves. *Annals of Mathematics 126* (1987), 649–673.