Short Note

A Fast $O(n^2)$ Division Algorithm for Multiple-Precision Floating-Point Numbers

KAZUFUMI OZAWA*

A fast $O(n^2)$ algorithm is derived for the division of multiple-precision floating-point numbers, where n is the number of digits in each of the numbers. This algorithm, which is a modification of the conventional pencil-and-paper technique, is as fast as the conventional $O(n^2)$ multiplication and 2.67 times faster than the algorithm based on the Newton method.

1. Introduction

Let X and Y be radix b multiple-precision floating-point (MPF) numbers

$$X=(-1)^{S_x}b^{E_x}(x_1b^{n-1}+x_2b^{n-2}+\cdots+x_n), \quad (1.1)$$
$$Y=(-1)^{S_y}b^{E_y}(y_1b^{n-1}+y_2b^{n-2}+\cdots+y_n),$$
$$S_x=S_y=0,\ 1,$$

where the digits x_i and y_i are integers satisfying

$$0\le x_i<b, \quad 0\le y_i<b,$$

and in particular

$$x_1\neq 0, \quad y_1\neq 0;$$

that is, the mantissas of X and Y are normalized. In what follows, we shall assume that the digits x_i and y_i are single-precision (SP) numbers. Without loss of generality we assume that

$$S_x=S_y=0,$$
$$E_x=E_y=0.$$

Then, using the conventional notation for radix b numbers, we have

$$X=(x_1x_2\cdots x_n.0)_b,\ Y=(y_1y_2\cdots y_n.0)_b, \quad (1.2)$$
$$Z=X/Y=(z_0.z_1z_2\cdots z_n\cdots)_b. \quad (1.3)$$

The usual pencil-and-paper algorithm for the division of MPF numbers is as follows:

begin
  $z_0:=X$ **div** Y; $r_0:=X$ **mod** Y;
  for $i:=1$ **to** n **do begin**
    $z_i:=b*r_{i-1}$ **div** Y; $r_i:=b*r_{i-1}$ **mod** Y (1.4)
  end
end.

Setting $br_{-1}=X$, we can find the relation equivalent to Algorithm (1.4)

$$z_i=\left\lfloor \frac{br_{i-1}}{Y} \right\rfloor, \quad r_i=br_{i-1}-z_iY, \quad i=0,\ 1,\ 2,\ \cdots,\ n. \quad (1.5)$$

It follows from Eq. (1.5) that

$$r_n=(\cdots(((X-z_0Y)b-z_1Y)b-z_2Y)b\cdots-z_{n-1}Y)b-z_nY \quad (1.6)$$
$$=b^nX-(b^nz_0+b^{n-1}z_1+\cdots+z_n)Y.$$

In Algorithm (1.4), the ith digit z_i of Z is calculated by the divide-and-correct technique, in which a certain estimate of z_i, say $\hat{z}_i$, is calculated from the first several digits of $b*r_{i-1}$ and Y, and is corrected to the exact value z_i. Various methods have been derived for obtaining a good estimate $\hat{z}_i$ that mininizes the number of corrections [2-6].

It should be noted, however, that the probability of a correction occuring is very small if the estimate $\hat{z}_i$ is properly selected, and moreover that the correction, if necessary, could be realized by at most two MPF additions or subtractions [1-3]. On the other hand, the calculation of r_i always requires n SP multiplications and n SP subtractions for all i, since in practice r_i is calculated from the expression $br_{i-1}-z_iY$, which contains one subtraction of MPF and one multiplication SP × MPF. Thus, in order to obtain a substantial reduction of the total time-complexity in the algorithm, it is necessary to reduce the cost of the calculation of r_i. The new algorithm we propose here has a reduced complexity and is as fast as the usual MPF multiplication algorithm.

2. A Fast Algorithm

Before considering the modified algorithm we define the truncated values of Y as follows:

*Department of Computer Science College of General Education, Tohoku University, Kawauchi, Aoba-ku, Sendai, Miyagi 980, Japan.

$$Y_i \equiv Y-(b^{i-1}y_{n-i+1}+b^{i-2}y_{n-i+2}+\cdots+y_n), \quad i=1, 2, \cdots, n-1. \tag{2.1}$$

If we set

$$Y_i = Y(1-c_i b^{-n+i}), \quad i=1, 2, \cdots, n-1, \tag{2.2}$$

then c_i satisfies the inequality

$$0 \le c_i < b. \tag{2.3}$$

By the use of Y_i and for some m we define the following new algorithm:

begin
  $\tilde{z}_0 := X$ **div** Y; $\tilde{r}_0 := X$ **mod** Y;
  for $i := 1$ **to** $m-1$ **do begin**
    $\tilde{z}_i := b*\tilde{r}_{i-1}$ **div** Y; $\tilde{r}_i := b*\tilde{r}_{i-1}$ **mod** Y (2.4)
  end;
  for $i := m$ **to** n **do begin**
    $\tilde{z}_i := b*\tilde{r}_{i-1}$ **div** Y_{i-m+1}; $\tilde{r}_i = b*\tilde{r}_{i-1}$ **mod** Y_{i-m+1}
  end
end.

This algorithm leads to the relations

$$\tilde{z}_i = z_i, \quad \tilde{r}_i = r_i, \quad i=0, 1, \cdots, m-1, \tag{2.5a}$$

$$\tilde{z}_i = \left\lfloor \frac{b\tilde{r}_{i-1}}{Y_{i-m+1}} \right\rfloor, \quad \tilde{r}_i = b\tilde{r}_{i-1} - \tilde{z}_i Y_{i-m+1}, \quad i=m, m+1, \cdots, n, \tag{2.5b}$$

where z_i and r_i are the quantities defined by Eq. (1.5). From Eq. (2.5) we have

$$\begin{aligned}\tilde{r}_n &= b^n X-(b^n\tilde{z}_0+b^{n-1}\tilde{z}_1+\cdots+b^{n-m+1}\tilde{z}_{m-1})Y \\ &\quad -(b^{n-m}\tilde{z}_m Y_1+b^{n-m-1}\tilde{z}_{m+1}Y_2+\cdots+\tilde{z}_n Y_{n-m+1}) \\ &= b^n X-\left\{\sum_{i=0}^{m-1} b^{n-i}\tilde{z}_i+\sum_{i=m}^{n} b^{n-i}\tilde{z}_i(1-c_{i-m+1}b^{-n+i-m+1})\right\} Y.\end{aligned} \tag{2.6}$$

The next theorem proves that Algorithm (2.4) gives a result that differs from $Z=X/Y$ by at most one unit of the nth place.

Theorem. *If we denote the quotient obtained from Algorithm (2.4) by*

$$Z' = (\tilde{z}_0.\tilde{z}_1 \cdots \tilde{z}_n)_b, \tag{2.7}$$

then for any m such that

$$0 < n-m+1 \ll b^{m-3}, \tag{2.8}$$

the error of Algorithm (2.4) is bounded by

$$|Z'-Z| < b^{-n}(1+o(1)). \tag{2.9}$$

Proof. It follows from Eq. (2.6) that the error is given by

$$Z'-Z = b^{-n}\left(-\tilde{r}_n Y^{-1}+\sum_{i=m}^{n} c_{i-m+1}\tilde{z}_i b^{-m+1}\right), \tag{2.10}$$

and using (2.3) and the inequalities

$$Y_{n-m+1} \le Y, \quad 0 \le \tilde{r}_n < Y_{n-m+1}, \tag{2.11}$$

we have

$$|Z'-Z| < b^{-n}[1+(n-m+1)b^{-m+3}]. \tag{2.12}$$

Therefore (2.9) is valid for any m that satisfies (2.8). Q.E.D.

Next we analyze the time-complexity of Algorithm (2.4), and compare it with that of the usual MPF multiplication algorithm. We define the time-complexity as the total number of SP multiplications and additions (subtractions) required for the algorithm; we ignore the cost of inexpensive operations such as the normalization of the mantissa or the releasing of carries. Since the lengths of Y and Y_{i-m+1} are n and $n+m-i-1$, respectively, the numbers of SP multiplications required are n for $\tilde{z}_i * Y$ and $n+m-i-1$ for $\tilde{z}_i * Y_{i-m+1}$. Therefore, in the overall computation of Algorithm (2.4) the number of SP multiplications is given by

$$D_d(n) \equiv mn+\sum_{i=m}^{n}(n+m-i-1) = \frac{n^2}{2}+O(n). \tag{2.13}$$

It follows that the total number of SP subtractions is also given by Eq. (2.13), since the number of SP subtractions required for one MPF subtraction is equal to the length of the MPF. It is easy to show that the time-complexity given by Eq. (2.13) is half of that of Algorithm (1.4).

We will now show that the complexity of Eq. (2.13) is as same as that of the usual MPF multiplication algorithm. Fox X and Y given by (1.2), the product $Z=X*Y$ is defined by

$$Z = X*Y = (z_1 z_2 \cdots z_{2n}.0)_b, \tag{2.14}$$

where the digits z_k are given by

$$z_k = \left(\sum_{i=1}^{l} x_i y_{k-i}+carry\right) \bmod b, \quad k=2, 3, \cdots, 2n, \tag{2.15}$$

$$l = \min\{n, k-1\},$$

$$z_1 = carry \bmod b.$$

In order to avoid the accumulation of round-off errors, we must compute the extra $q(>0)$ digits, nemely, z_{n+1}, z_{n+2}, $\cdots$, z_{n+q}, although in floating-point arithmetic only z_1, $\cdots$, z_n are necessary. Consequently, the numbers of SP multiplications and SP additions are given by

$$M(n) \equiv \sum_{k=1}^{n+1}(k-1)+\sum_{k=n+2}^{n+q} n = \frac{n^2}{2}+O(n), \tag{2.16}$$

for any q independent of n. This means that the time-complexity of Algorithm (2.4) is as same as that of the MPF multiplication, except for the $O(n)$ terms.

Next we will compare the complexity of (2.13) with that of the division algorithm [1] based on the Newton iteration

$$u_{+1}=u_i+\Delta(u_i), \quad i=0, 1, \cdots, \qquad (2.17)$$
$$\Delta(u_i)=u_i(1-Yu_i).$$

Using (2.17) with an accurate initial value u_0 such that the method converges quadratically, we will have a sufficiencly accurate approximation to Y^{-1} after several iterations. We then multiply it by X to obtain a sufficiently accurate approximation to $Z=X/Y$. We consider the total complexity of thie algorithm.

Let u_i be an approximation to Y^{-1} that is correct up to the n_ith digit of Y^{-1}. Then the relation

$$|\Delta(u_i)|/|Y^{-1}| \approx |u_i-Y^{-1}|/|Y^{-1}| \approx b^{-n_i}. \qquad (2.18)$$

holds. In view of this relation, it is sufficient to compute $\Delta(u_i)$ up to the n_ith digit in order to get the next iterate u_{i+1} being correct up to the $2n_i(=n_{i+1})$th digit. To compute the upper n_i digits of $\Delta(u_i)$, we must compute Yu_i and $u_i(1-Yu_i)$ up to the $2n_i$th digit and up to the n_ith digit, respectively. If n is an integer such that the ratio n/n_0 can be represented exactly by an integral power of 2, the time-complexity of the method, that is, the number of SP multiplications and additions (subtractions), is given by

$$D_N(n) \equiv \sum_{i=0}^{p-1}\left\{\frac{1}{2}(2n_i)^2+\frac{1}{2}n_i^2\right\}+\frac{1}{2}n^2+O(n)$$
$$\approx \frac{4}{3}n^2+O(n)=2.67D_d(n), \qquad (2.19)$$

where p is an integer given by $p=\log_2(n/n_0)$.

3. Numerical Experiments

We performed dividions using actual random data to confirm the validity of our new algorithm. The radix b for the MPF is 10 and the MPF number is represented in these computations as an array of integers. The first and second cells of the array contain the sign and the exponent of the number, respectively. The remaining cells contain integers $<10^8$ to represent the mantissa. For example, the number

$$-0.1234567890123456789000\cdots\times 10^8$$

is represented by the array -1, 8, 12345678, 90123456, 78900000, 0, $\cdots$, 0. The values of n in the experiments are 1024, 2048, $\cdots$, 32768, while m in Algorithm (2.4) is 24. The computer used for these computations was the ACOS-2020 at the Computer Center of Tohoku University. The experimental results in Tables 1 and 2 show that our new algorithm is as fast as the multiplication, and gives results whose accuracy is comparable to *machine* ε.

Table 1 CPU-time for the division algorithms.

	New algorithm		Newton method		Multiplication	
n	time (ms)	ratio	time (ms)	ratio	time (ms)	ratio
1024	9	1.29	52	7.43	7	1
2048	34	1.26	110	4.07	27	1
4096	129	1.17	325	2.95	110	1
8192	507	1.17	1172	2.69	435	1
16384	2028	1.17	4535	2.62	1734	1
32768	8174	1.18	18130	2.61	6952	1

Table 2 Relative error of the algorithms.

n	New algorithm	Newton
1024	5.44E- 1024	-7.61E- 1025
2048	5.97E- 2048	-1.77E- 2048
4096	2.94E- 4096	-1.70E- 4096
8192	3.96E- 8192	-2.23E- 8192
16384	1.77E-16384	-1.33E-16384
32768	-7.10E-32769	-7.10E-32769

Acknowledgement

The author is grateful to Dr. K. Kaino of the Sendai National College of Technology for his helpful suggestions.

References

1. BRENT, R. P. Fast Multiple-Precision Evaluation of Elementary Functions, *J. ACM* **23** (1976), 242-251.
2. KNUTH, D. E. *The Art of Computer Programming, Seminumerical Algorithms, 2nd ed.*, Addison-Wesley, 1981.
3. KRISHNAMURTHY, E. V. On a Divide-and-Correct Method for Variable-Precision Division, *Comm. ACM*, **8** (1965), 179-181.
4. ONAKA, K. and YASUI, H. Multiple-precision Arithmetic with Error Estimation (in Japanese), *Trans. IPS Japan*, **15** (1974), 110-117.
5. SEMBA, I. An Algorithm for Division of Large Integers, *J. Inf. Process.*, **9** (1986), 145-147.
6. TODA, H. and ONO, H. On Algorithms for Multiple-Precision Division (in Japanese), *Bul. Electrotech. Lab.* **42** (1978), 66-71

(Received October 26, 1990)