Analysis and Improvement of Kahan's Summation Algorithm

KAZUFUMI OZAWA*

This paper shows an example in which Kahan's algorithm for summation is not at all effective, and consider the reasons for the uneffectiveness. Next, this paper proposes a new algorithm similar to Kahan's. Application of this new algorithm to the same and other examples show that the algorithm improves the results over Kahan's. Error analyses of the two algorithms show that, in both algorithms, the quantities approximating the accumulated errors can be obtained in the course of computations. In particular, this quantities of the improved algorithm approximates the error more precisely than that of Kahan's. The cpu time of the improved algorithm is about 1.4 times that of Kahan's.

1. Introduction

Various algorithms [1–4] have been derived for reducing the round-off error in summation of floating-point numbers, and some algorithms are successfully applied to the numerical integration of ordinary differential equations [4].

In Linz's [2] and Wolfe's [3] algorithms, a floating-point number is added in such a manner that its magnitude does not differ much from that of the partial sum, so that the round-off error in addition is reduced. However, these algorithms require large storage.

On the other hand, Kahan [1] has derived an algorithm in which the round-off error in addition is estimated and corrected. The algorithm requires smaller storage. Until now, only the results of research showing that Kahan's algorithm is more accurate have been presented (see [5], [6], and [7]). However, whether the algorithm is accurate or not is unknown, since no one has analyzed the error.

In this paper, we present an example in which Kahan's algorithm produces the same result as that produced by the simpler and more conventional algorithm—In Sec. 2, we will call it Algorithm 0. By considering the error in this example, we improve the algorithm and analyze the error of the improved one. The analysis shows that, in the improved algorithm, we can always obtain a quantity which approximates the accumulated error precisely.

2. Kahan's Algorithm

We begin with the simpler algorithm, say Algorithm 0, for the summation of floating-point numbers X_i, $i=1, 2, \cdots, n$.

Algorithm 0

S0 $S_0:=0$

for $i:=1$ to n

S1 $S_i:=S_{i-1}+X_i$

If the sum were computed by this algorithm, then, in almost all cases, the round-off error would increase, as i increases. In order to prevent this, Kahan [1] has derived an algorithm, which estimates and corrects the error in addition. The algorithm is as follows:

Algorithm 1

K0 $Q_0:=0, S_0:=0$

for $i:=1$ to n

K1 $V_i:=X_i-Q_{i-1}$

K2 $S_i:=S_{i-1}+V_i$

K3 if $|S_{i-1}|<|V_i|$, exchanges S_{i-1} and V_i

K4 $Q_i:=(S_i-S_{i-1})-V_i$

In [1], Kahan has showed a FORTRAN implementation of the algorithm without exchanging S_{i-1} and V_i. However, Shimizu et al. [8] and Linnainmaa [9] have proved that it is necessary to exchange S_{i-1} and V_i, if $|S_{i-1}|<|V_i|$, to make the algorithm more accurate. In the following, we will restrict ourselves to Algorithm 1 not to the original one.

In Algorithm 1, the round-off error Q_i arising at K2 is calculated and stored to correct the sum S_{i+1}. In order to perform this correction successfully, it is necessary to use the following type of computer (see [8]):

Table 1 Specification of computer to be used.

base	: any
representation of negative numbers	: sign and magnitude representation
accumulator	: single-precision accumulator with guard digit
mode of rounding	: chopping

Even if we use the above type of computer, however, it is possible that the algorithm produces an inaccurate result, because of that, in Algorithm 1, Q_i does not express the sum of the errors at K1 and K2, in spite of that the total local error at step i is the sum of the both

*Sendai Radio Technical College, Miyagi, Japan.

errors. We consider an extreme case that $S_{i-1}+V_i$ at K2 is performed without error, but X_i-Q_{i-1} at K1 is performed with error. In this case $Q_i=0$, and consequently at the next step the quantity S_{i+1} remains uncorrected. In the following, we show an example in which this extreme case occurs frequently.

Example 1

We compute $S_n=1+\alpha-1+\alpha+1+\alpha-1\cdots-1+\alpha$ by Algorithm 0 and 1, where $\alpha\neq 0$ is a small number such that $1\pm i\alpha=1$, for $i=1, 2, \cdots, n$ on the computer. The results are shown in Table 2 and 3.

Table 2 Result by Algorithm 0.

i	X_i	S_{i-1}	S_i	s_i (exact)	Error S_i-s_i
1	1	0	1	1	0
2	α	1	1	$1+\alpha$	$-\alpha$
3	-1	1	0	α	$-\alpha$
4	α	0	α	2α	$-\alpha$
5	1	α	1	$1+2\alpha$	-2α
6	α	1	1	$1+3\alpha$	-3α
7	-1	1	0	3α	-3α
8	α	0	α	4α	-3α

Table 3 Result by Algorithm 1.

i	X_i	Q_{i-1}	V_i	S_{i-1}	S_i	Q_i	s_i (exact)	Error S_i-s_i
1	1	0	$1\to 0$	$0\to 1$	1	0	1	0
2	α	0	α	1	1	$-\alpha$	$1+\alpha$	$-\alpha$
3	-1	$-\alpha$	-1	1	0	0	α	$-\alpha$
4	α	0	$\alpha\to 0$	$0\to\alpha$	α	0	2α	$-\alpha$
5	1	0	$1\to\alpha$	$\alpha\to 1$	1	$-\alpha$	$1+2\alpha$	-2α
6	α	$-\alpha$	2α	1	1	-2α	$1+3\alpha$	-3α
7	-1	-2α	-1	1	0	0	3α	-3α
8	α	0	$\alpha\to 0$	$0\to\alpha$	α	0	4α	-3α
9	1	0	$1\to\alpha$	$\alpha\to 1$	1	$-\alpha$	$1+4\alpha$	-4α
10	α	$-\alpha$	2α	1	1	-2α	$1+5\alpha$	-5α
11	-1	-2α	-1	1	0	0	5α	-5α
12	α	0	$\alpha\to 0$	$0\to\alpha$	α	0	6α	-5α

The S_i's of Algorithm 1 always have the same values with those of Algorithm 0, i.e., Algorithm 1 is uneffective in this example. This means that the correction of S_i by Q_{i-1} is not done at all.

3. Improved Algorithm

In this section, we propose an improved algorithm, in which not only the error at $S_{i-1}+V_i$ but also the error at X_i-Q_{i-1} are considered. The algorithm is as follows:

Algorithm 2

I0 $Q_0:=0$, $S_0:=0$
for $i:=1$ to n
I1 $V_i:=X_i-Q_{i-1}$
I2 $S_i:=S_{i-1}+V_i$
I3 if $|X_i|<|Q_{i-1}|$, exchange X_i and $(-Q_{i-1})$
I4 $U_i:=(V_i-X_i)+Q_{i-1}$
I5 if $|S_{i-1}|<|V_i|$, exchange S_{i-1} and V_i
I6 $W_i:=(S_i-S_{i-1})-V_i$
I7 $Q_i:=U_i+W_i$

Example 2

We will apply this algorithm to the same calculation as Example 1. As in Example 1, α is a small number such that $1\pm i\alpha=1$ on the computer. The result is shown in Table 4.

Notice in Example 2 that Q_i is always equal to the accumulated error at each step. This means that we can estimate the accumulated error by using Q_i. Next we consider an example occurring in statistical problems.

Example 3

Let,

$$\text{(A)}\quad S^1_{10000}=\sum_{i=1}^{10000} X_i$$

$$\text{(B)}\quad S^2_{10000}=\sum_{i=1}^{10000} X_i^2$$

where X_i's are the Gaussian random numbers with mean 0 and variance 1. All the results by Algorithm 0, 1, and 2 are shown in Table 5.

Notice that, in each of (A) and (B), the error of Algorithm 2 is considerably smaller than those of Algorithm 0 and 1, and moreover that Q_{10000} of Algorithm 2 agrees with the accumulated error very well.

In this example, all the calculations are performed by a COSMO-700S computer, the specification of which is the same with that of Table 1, and the exact value is cal-

Table 4 Result by Algorithm 2.

i	X_i	Q_{i-1}	V_i	S_{i-1}	S_i	U_i	W_i	Q_i	s_i (exact)	Error S_i-s_i
1	1		$1\to 0$	$0\to 1$	1	0	0	0	1	0
2	α	0	α	1	1	0	$-\alpha$	$-\alpha$	$1+\alpha$	$-\alpha$
3	-1	$-\alpha$	-1	1	0	$-\alpha$	0	$-\alpha$	α	$-\alpha$
4	α	$-\alpha$	$2\alpha\to 0$	$0\to 2\alpha$	2α	0	0	0	2α	0
5	1	0	$1\to 2\alpha$	$2\alpha\to 1$	1	0	-2α	-2α	$1+2\alpha$	-2α
6	$\alpha\to 2\alpha$	$-2\alpha\to-\alpha$	3α	1	1	0	-3α	-3α	$1+3\alpha$	-3α
7	-1	-3α	-1	1	0	-3α	0	-3α	3α	-3α
8	$\alpha\to 3\alpha$	$-3\alpha\to-\alpha$	$4\alpha\to 0$	$0\to 4\alpha$	4α	0	0	0	4α	0
9	1	0	$1\to 4\alpha$	$4\alpha\to 1$	1	0	-4α	-4α	$1+4\alpha$	-4α
10	$\alpha\to 4\alpha$	$-4\alpha\to-\alpha$	5α	1	1	0	-5α	-5α	$1+5\alpha$	-5α
11	-1	-5α	-1	1	0	-5α	0	-5α	5α	-5α
12	$\alpha\to 5\alpha$	$-5\alpha\to-\alpha$	$6\alpha\to 0$	$0\to 6\alpha$	6α	0	0	0	6α	0

Table 5 Results of Example 3.

(A) Sum of the Gaussian random numbers.				
	Algorithm 0	Algorithm 1	Algorithm 2	Exact
S_{10000}	−0.2346869E 2	−0.2352652E 2	−0.2352655E 2	−0.2352655E 2
Error	0.5786772E-1	0.3691111E-4	0.6393529E-5	
Q_{10000}		0.1120567E-4	0.6393529E-5	
(B) Sum of the squares of the Gaussian random numbers.				
	Algorithm 0	Algorithm 1	Algorithm 2	Exact
S_{10000}	0.1000155E 5	0.1001326E 5	0.1001326E 5	0.1001326E 5
Error	−0.1171869E 2	−0.3850549E-2	−0.3850549E-2	
Q_{10000}		−0.2847075E-2	−0.3850540E-2	

culated by double-precision arithmetic. The Gaussian random numbers are generated by Box-Muller's method.

4. Error Analysis

The preceding examples show that the results of Algorithm 2 have smaller errors than those of Algorithm 0, and 1, and moreover that the quantity Q_i of Algorithm 2 approximates the accumulated round-off error almost precisely. In this section, we shall derive relations between Q_i and the accumulated errors for both of the algorithms.

4.1 Error Analysis of Algorithm 1

We will first introduce the symbols $\oplus$ and $\ominus$ to denote floating-point addition and subtraction, in order to distinguish the approximate operations from the true ones. By using these symbols and omitting the exchanging process K3, we can rewrite Algorithm 1 as follows:

Algorithm 1

$$\begin{aligned} &S_0=0,\ Q_0=0\\ &\text{for } i:=1 \text{ to } n\\ &V_i=X_i\ominus Q_{i-1}\\ &S_i=S_{i-1}\oplus V_i\\ &Q_i=(S_i\ominus S_{i-1})\ominus V_i \end{aligned} \tag{4.1}$$

We first show that Q_i denotes the error in $S_{i-1}\oplus V_i$, exactly. As Shimizu et al. [8] have proved, if $|A|\geq|B|$, then the relation,

$$A\oplus B=(A+B)+(((A\oplus B)\ominus A)\ominus B) \tag{4.2}$$

holds, provided that the computer specified in Table 1 is used and that no exponent overflow and underflow occurs. Applying (4.2) to (4.1), we have

$$\begin{aligned} S_i=S_{i-1}\oplus V_i&=(S_{i-1}+V_i)+(((S_{i-1}\oplus V_i)\ominus S_{i-1})\ominus V_i)\\ &=S_{i-1}+V_i+((S_i\ominus S_{i-1})\ominus V_i)\\ &=S_{i-1}+V_i+Q_i,\ i=1,2,\cdots,n, \end{aligned} \tag{4.3}$$

where $|S_{i-1}|\geq|V_i|$ is assumed. It follows, from (4.3), that Q_i is the error arising at $S_{i-1}\oplus V_i$:

From the relation (4.3), we will derive the expression for estimating the accumulated round-off error. Let s_i be the exact sum of $X_1, X_2,\cdots, X_i$. Then, the accumulated error $R_i=S_i-s_i$ satisfies that

$$\begin{aligned} R_i&=S_i-s_i\\ &=S_{i-1}+V_i+Q_i-s_i\\ &=S_{i-1}+V_i+Q_i-s_{i-1}-X_i\\ &=R_{i-1}+V_i+Q_i-X_i,\\ &i=1,2,\cdots,n,\\ &R_0=S_0-s_0=0. \end{aligned} \tag{4.4}$$

Summing up this relation from $i=1$ to n, we have

$$\begin{aligned} R_n&=\sum_{i=1}^{n}(V_i+Q_i-X_i)\\ &=\sum_{i=1}^{n}[(X_i\ominus Q_{i-1})+Q_i-X_i]. \end{aligned} \tag{4.5}$$

Let ε_i be the relative error in $X_i\ominus Q_{i-1}$, i.e.,

$$X_i\ominus Q_{i-1}=(X_i-Q_{i-1})(1+\varepsilon_i),\ i=1,2,\cdots,n,$$

then (4.5) reduces to

$$R_n=\sum_{i=2}^{n}(X_i-Q_{i-1})\varepsilon_i+Q_n, \tag{4.6}$$

where $\varepsilon_1=0$ is used.

From (4.6), it is easily shown that

$$\begin{aligned} |R_n-Q_n|&\leq\sum_{i=2}^{n}(|X_i|+|Q_{i-1}|)|\varepsilon_i|\\ &\leq(n-1)(L+M)\rho, \end{aligned} \tag{4.7}$$

where

$$L\equiv\max_{2\leq i\leq n}|X_i|,\ M\equiv\max_{2\leq i\leq n}|Q_{i-1}|,$$
$$\rho\equiv\max_{2\leq i\leq n}|\varepsilon_i|.$$

The bound ρ is given by

$$\rho=b^{-t+1}, \tag{4.8}$$

if the length of the guard digit is greater than 1, where b is the base of the floating-point system and t is the length of mantissa (see Sterbentz [11]).

We can conclude, from (4.7), the accumulated error R_n can be estimated by Q_n precisely, since, in general, ρ is a small number.

4.2 Error Analysis of Algorithm 2

As before, we first rewrite Algorithm 2 by using $\oplus$, $\ominus$:

Algorithm 2

$$S_0=0,\ Q_0=0$$

$$\begin{aligned}&\text{for } i:=1 \text{ to } n\\&V_i=X_i\ominus Q_{i-1}\\&S_i=S_{i-1}\oplus V_i\\&U_i=(V_i\ominus X_i)\oplus Q_{i-1}\\&W_i=(S_i\ominus S_{i-1})\ominus V_i\\&Q_i=U_i\oplus W_i\end{aligned}\tag{4.9}$$

In the following, we show that, in (4.9), Q_i is an approximation of the local error for the ith step. At the ith step, we want to compute $S_{i-1}+(X_i-Q_{i-1})$ exactly, but, in fact, S_i is obtained. Therefore,

$$r_i\equiv S_i-(S_{i-1}+X_i-Q_{i-1}),\ i=1, 2,\cdots, n$$

means the local round-off error for the ith step. The r_i satisfies that

$$r_i=U_i+W_i,\ i=1, 2,\cdots, n,\tag{4.10}$$

since, from (4.2) and (4.9),

$$\begin{aligned}S_i=S_{i-1}\oplus V_i&=(S_{i-1}+V_i)+((S_i\ominus S_{i-1})\ominus V_i)\\&=S_{i-1}+V_i+W_i,\ i=1, 2,\cdots, n\end{aligned}\tag{4.11}$$

and

$$\begin{aligned}V_i=X_1\oplus(-Q_{i-1})&=X_i-Q_{i-1}+((V_i\ominus X_i)\oplus Q_{i-1})\\&=X_i-Q_{i-1}+U_i,\ i=1, 2,\cdots, n\end{aligned}\tag{4.12}$$

hold, where $|S_{i-1}|\geq|V_i|$ and $|X_i|\geq|Q_{i-1}|$ are assumed. By (4.10), we find that $Q_i=U_i\oplus W_i$ is an approximation of r_i. Next we show that Q_i is also an approximation of the accumulated round-off error.

The accumulated round-off error $R_i=S_i-s_i$ satisfies that

$$\begin{aligned}R_i&=S_i-s_i\\&=(S_{i-1}+V_i+W_i)-(s_{i-1}+X_i),\\&=R_{i-1}+r_i-Q_{i-1},\end{aligned}\tag{4.13}$$

where (4.11) and (4.12) are used. Here, we introduce the relative error δ_i in $U_i\oplus W_i$, i.e.,

$$\begin{aligned}Q_i&=U_i\oplus W_i\\&=(U_i+W_i)(1+\delta_i).\end{aligned}\tag{4.14}$$

As before $|\delta_i|\leq\rho$. Using δ_i and neglecting the higher order terms of δ_i, we may reduce (4.13) to

$$\begin{aligned}R_i&=R_{i-1}+(1+\delta_i)^{-1}Q_i-Q_{i-1}\\&\simeq R_{i-1}+Q_i-Q_{i-1}-Q_i\delta_i,\\&i=1, 2,\cdots, n,\end{aligned}\tag{4.15}$$

and, from (4.9), we have

$$\begin{aligned}&R_0=R_1=0,\ \delta_1=0,\\&Q_0=Q_1=0.\end{aligned}\tag{4.16}$$

Summing up (4.15) from $i=1$ to n, we have

$$\begin{aligned}R_n&=-\sum_{i=2}^{n}Q_i\delta_i+Q_n\\&=-\sum_{i=2}^{n}Q_i\delta_i+Q_n(1-\delta_n)\\&\simeq-\sum_{i=2}^{n-1}Q_i\delta_i+Q_n.\end{aligned}\tag{4.17}$$

From this, the relation

$$\begin{aligned}&|R_n-Q_n|\leq(n-2)M\rho,\\&M\equiv\max_{2\leq i\leq n-2}|Q_i|\end{aligned}\tag{4.18}$$

is easily derived.

Next we derive the bound M for $|Q_i|$. From (4.11), (4.12), and (4.14), we have

$$\begin{aligned}Q_i&=(U_i+W_i)(1+\delta_i)\\&=[(V_i-(X_i-Q_{i-1}))+(S_i-(S_{i-1}+V_i))](1+\delta_i).\end{aligned}\tag{4.19}$$

Here, we define the relative errors in V_i and S_i:

$$\begin{aligned}V_i&=X_i\ominus Q_{i-1}=(X_i-Q_{i-1})(1+\alpha_i)\\S_i&=S_{i-1}\ominus V_i=(S_{i-1}-V_i)(1+\beta_i),\end{aligned}\tag{4.20}$$

where $|\alpha_i|\leq\rho$, $|\beta_i|\leq\rho$.
Using this, we have

$$\begin{aligned}Q_i&=((X_i-Q_{i-1})\alpha_i+(S_{i-1}-V_i)\beta_i)(1+\delta_i)\\&=(V_i(1+\alpha_i)^{-1}\alpha_i+S_i(1+\beta_i)^{-1}\beta_i)(1+\delta_i)\\&=V_i\alpha_i+S_i\beta_i+0(\rho^2).\end{aligned}\tag{4.21}$$

and therefore,

$$\begin{aligned}|Q_i|&\leq\rho(|V_i|+|S_i|)+0(\rho^2)\\&\leq\rho(|S_{i-1}|+|S_i|)+0(\rho^2)\\&\lesssim 3\rho|S_{i-1}|,\end{aligned}\tag{4.22}$$

where

$$|S_i|=|S_{i-1}\oplus V_i|\lesssim|S_{i-1}|+|V_i|\leq 2|S_{i-1}|$$

is used. Thus, we have

$$\begin{aligned}M&=\max_{2\leq i\leq n-1}|Q_i|\\&=3\rho\max_{1\leq i\leq n-2}|S_i|,\end{aligned}\tag{4.23}$$

and

$$|R_n-Q_n|\leq 3(n-2)\max_{1\leq i\leq n-2}|S_i|\rho^2.\tag{4.24}$$

Because of (4.24), the right-hand side of (4.18) is considerably improved over that of (4.7). Therefore, it is expected that the accumulated error in Algorithm 2 can be estimated in term of Q_n more precisely than in Algorithm 1.

Table 6 Cpu time ratio of various algorithms to Algorithm 0.

Algorithm 0	Algorithm 1	Algorithm 2	Wolfe	Linz	Algorithm 0 (double)
1.00	9.95	14.0	13.4	2.00	2.65

5. Proper Rounding Arithmetic

Until now, we have discussed only the case in which the computer choppes the lower digits. But a computer which rounds the lower digits properly is also frequently used. If the two numbers A and B are added by the computer, then the following relations holds [4], [10]:

$$A+B=(A\oplus B)-(A'\ominus A)\oplus(B''\ominus B)$$
$$A'=(A\oplus B)\ominus B,\ B''=(A\oplus B)\ominus A', \qquad (5.1)$$

where $\oplus$ and $\ominus$ have the same meanings as before. If we rewrite the step K4 of Algorithm 1 by using (5.1), and delete the exchanging process K3, then Algorithm 1 is changed into one suitable for proper rounding arithmetic. The algorithm derived by this revision is called Mϕller's algorithm [4]. In the same manner, we can revise Algorithm 2 so that it is suitable for proper rounding arith metic. By these revisions of the two algorithms, no change in the discussion of Sec. 4.1 and 4.2 is required, except for the value of ρ, i.e., ρ must be halved.

Finally, we compare cpu time of the various algorithms (see Table 6). In these experiments, a COSMO-700S computer was used, and Wolfe's algorithm and Linz's algorithm were performed by using the codes proposed in [6] and [12], respectively.

Acknowledgment

The author would like to thank Dr. Takeda and Dr. Abe for their helpful suggestions.

References

1. KAHAN, W. Further Remarks on Reducing Truncation Errors, *CACM* **8**, (1965), 40.
2. LINZ, P. Accurate Floating-point Summation, *CACM* **13**, (1970), 361–362.
3. WOLFE, J. M. Reducing Truncation Errors by Programming, *CACM* 7, (1964), 355.
4. MØLLER, O. Quasi Double-precision in Floating-point Addition, *BIT* **5**, (1965), 37–50.
5. GREGORY, J. A Comparison of Floating-point Summation Methods, *CACM* **15**, (1972), 838.
6. YAMASHITA, S. Fudosyosuten-Enzan no gosa, *bit Rinji Zokan*, (1975), (in Japanese), 14–26.
7. USHIJIMA, K. and ASHIDA, T. Fudosyosuten-Su no sowa no keisan-ho no hikaku, *Surikagaku Kokyuroku* **215**, (1974), (in Japanese), 75–86.
8. SHIMIZU, T. and OOHASHI, T. Rounding Errors in Floating-point Addition, *TRU Mathematics* **11**, (1975), 41–50.
9. LINNAINMAA, S. Analysis of Some Known Methods of Improving the Accuracy of Floating-point Sums, *BIT* **14**, (1974), 167–202.
10. KNUTH, D. E. The Art of Computer Programming, Vol. 2, Addison-Wesley, Reading Mass., (1973).
11. STERBENTZ P.H. Floating-point Computation, Prentice-Hall, Englewood Cliffs, N.J., (1974).
12. TOGAWA, H. Gosa-Kaiseki no kiso, Science Sha, Tokyo, (in Japanese), (1974).

(Received February 28, 1983; revised July 18, 1983)