How Children Accumulate Numbers – or: Why We Need a Fifth Floating-Point Operation

Tim Hoff

When John was in fourth grade, he learned how to accumulate a sequence of numbers. This is one of the problems he had to do for his homework:

At a bank, the following incoming (+) and outgoing (−) payments are registered on a certain day. Compute the total sum of these transactions:

1. Customer A pays for his house: $-1,355,100$ \$
2. Government revenue: $+287,312$ million \$
3. Customer B pays for his car: $-15,185,$ \$
4. Government expenditures: $-144,239$ million \$
5. Customer C sold his home: $+444,998$ \$
6. Customer D buys a house lot: $-425,187$ \$
7. Interest payment of customer E: $+24,998$ \$
8. Salary of customer F: $+4,972.29$ \$
9. Interest payment of customer G: $-5,112$ \$
10. Government expenditures: $-143,072$ million \$
11. Garage sale of customer H: $+1,417.49$ \$
12. Customer J sold his farm: $+2,425,700$ \$

John lined up all 12 summands as in the table below (where the i-th transaction is denoted by a_i). Then he added all the positive and all the negative numbers and finally took the difference giving the total sum of $+2,101,501.78$ \$.

a_1	:	−	1 355 100.00
a_2	:	+	287 312 000 000.00
a_3	:	−	15 185.00
a_4	:	−	144 239 000 000.00
a_5	:	+	444 998.00
a_6	:	−	425 187.00
a_7	:	+	24 998.00
a_8	:	+	4 972.29
a_9	:	−	5 112.00
a_{10}	:	−	143 072 000 000.00
a_{11}	:	+	1 417.49
a_{12}	:	+	2 425 700.00
			2 101 501.78

Ten years later, John was studying at a renowned university. In his Numerical Analysis course, the professor talked about Scientific Computing and solving huge real world problems. Floating-point arithmetic is the fastest way of doing calculations, the professor said. IEEE arithmetic is an optimal way to perform floating-point arithmetic. Every floating-point operation is as accurate as possible. The professor talked a lot about additional features of the IEEE arithmetic standard which make it particularly intelligent. The IEEE number system includes plus and minus infinity, and one can even calculate with objects called NaNs (Not a Number). There are also two zeros in this number system, a $+0$ and a -0, which are different.

John was very impressed. Since he was at a renowned university, he believed everything the professor said. He bought a pocket calculator which was equipped with an optimal floating-point arithmetic, and from then on, did all his calculations on this calculator. He was absolutely convinced that this was the quickest and best way of doing calculations.

Another ten years later, John's daughter Ann was in fourth grade, and she had to do the same problem. She did the calculation by hand, just as her father had done twenty years before. She obtained the same result her father had obtained then, which of course he did not remember after twenty years. When John looked over Ann's homework that evening, he didn't recognize the old problem, but he was intrigued by it (and, to tell the truth, he wanted to double check his daughter's calculation). Out of habit, John used his old calculator. It used a decimal floating-point arithmetic with a significand (mantissa) of 8 digits. Since none of the summands in the problem contain more than six significant digits, John did not foresee any problems. He did the calculation using the algorithm:

$$c_1 := a_1; \quad \text{for } i := 2 \text{ to } 12 \text{ do } c_i := c_{i-1} + a_i;$$

as shown in the table below. The symbols $\oplus$ and $\ominus$ indicate that floating-point operations are performed where the result of an operation is always rounded to 8 decimal digits.

$$\begin{aligned}
c_1 &:= -1\,355\,100\\
c_2 &:= c_1 \oplus 287\,312 \cdot 10^6 = 287\,310\,64 \cdot 10^4\\
c_3 &:= c_2 \ominus 15\,185 = 287\,310\,62 \cdot 10^4\\
c_4 &:= c_3 \ominus 144\,239 \cdot 10^6 = 143\,071\,62 \cdot 10^4\\
c_5 &:= c_4 \oplus 444\,998 = 143\,072\,06 \cdot 10^4\\
c_6 &:= c_5 \ominus 425\,187 = 143\,071\,63 \cdot 10^4\\
c_7 &:= c_6 \oplus 24\,998 = 143\,071\,65 \cdot 10^4\\
c_8 &:= c_7 \oplus 4\,972.29 = 143\,071\,65 \cdot 10^4\\
c_9 &:= c_8 \ominus 5\,112 = 143\,071\,64 \cdot 10^4\\
c_{10} &:= c_9 \ominus 143\,072 \cdot 10^6 = -360 \cdot 10^3\\
c_{11} &:= c_{10} \oplus 1\,417.49 = -358\,582.51\\
c_{12} &:= c_{11} \oplus 2\,425\,700 = 2\,067\,117.5
\end{aligned}$$

Comparing his result with Ann's, John realized that they were different. There was a difference of more than 34 thousand dollars! Ann must have made a mistake, John thought. He checked Ann's calculation, but could not find anything wrong with it.

What was going on here? Was his old little calculator suddenly suffering of old age? John repeated the same calculation on the PC which he had bought earlier that year. Using binary IEEE single precision arithmetic, he obtained 2, 117, 869.5 which also deviates substantially from Ann's result. He checked this result on the workstation in his office and on a parallel computer which were both equipped with IEEE arithmetic and always got the same answer.

Under normal circumstances, John would have been convinced that this was the correct result. However, he could not find any fault with Ann's hand calculation either. So out of curiosity, John tried the "ancient" mechanical calculator which his grandfather, who had been an engineer, had used 75 years ago. To his surprise the old machine produced exactly the answer that Ann had found. Thoroughly confused and somewhat irritated, John carefully repeated the accumulation by hand once more. Ann's result was confirmed. It was now beyond doubt that Ann had been right all along. It dawned upon John that the floating-point process in itself was at fault and led to wrong results. IEEE arithmetic made the bank richer by more than 16 thousand dollars, and his old pocket calculator made it poorer by more than 34 thousand dollars after only 12 transactions.

John was shocked and upset. For ten years he had done all his calculations, even his income taxes, with this calculator. Poor Ann! She was thoroughly confused by her father's investigations. The only thing she knew was that she wouldn't be using her dad's calculator during the exam! John decided to write to the manufacturer and complain. The answer he got was simple. It said: "Dear Sir, you should be aware of the fact that **floating-point accumulation** can easily lead to incorrect results even if every single operation is maximally accurate according to the IEEE arithmetic standard."

Dear reader: Did you know that modern computers performing floating-point arithmetic of the best quality may fail in simple accumulations which every child in fourth grade can do correctly within a few minutes? As a matter of fact, John had worked much harder and longer than Ann to get the correct answer! Why are

computers still built in this way and why do we purchase them? Of course, John could have obtained the correct answer with double precision in this case. But in more critical problems, double precision as well as extended precision will fail for similar reasons. The situation is particularly dangerous because the computer does not indicate in any way that the result of a floating-point accumulation may be incorrect. John was able to detect the error only because he had Ann's correct answer.

There is only one natural and easy way out of this situation: IEEE arithmetic has to be extended by a fifth operation, namely the accumulation of floating-point numbers or of simple products of such numbers into a fixed-point register. This is how children do it in school, how old-fashioned calculators did it and how accounting is normally done. It is the only reasonable way! **Fixed-point accumulation** is error-free even if thousands of positive and negative summands are added! An error analysis is never necessary! The technical realization is easy. Furthermore, fixed-point accumulation can be performed faster than floating-point accumulation because it is simpler. It is high time to require its realization from all computer manufacturers! It is most unreasonable to use an unreliable method where an absolutely safe and reliable technique is available. It is a sad fact that the situation on existing supercomputers is even worse than on ordinary sequential computers.

If the accumulation of floating-point numbers and of products of such numbers into a fixed-point register is introduced as a fifth floating-point operation, many unnecessary errors that appear in floating-point computations can be avoided. Furthermore, errors that occur during a floating-point computation can often be resolved by defect correction techniques. This requires accurate accumulation.