

# Notes on the accuracy and stability of naive summation

S. G. Johnson, MIT Course 18.335

Created Fall 2019; updated February 22, 2021

## 1 Summation: Order matters

Consider the summation function  $f(x) = \sum_{i=1}^n x_i$  for vectors  $x \in \mathbb{R}^n$  or  $x \in \mathbb{F}^n$  ( $n$  real numbers). When we implement an approximate version  $\tilde{f}(x)$  of this sum on a computer, using  $\oplus$  (floating-point addition) rather than  $+$  (exact addition), there are many possible algorithms, and different algorithms will have different roundoff errors.

A key fact to remember is that  $\oplus$  is commutative but *not associative*. In particular, consider  $\frac{1}{\epsilon_{\text{machine}}^2} + \frac{-1}{\epsilon_{\text{machine}}^2} + 1 = 1$ . In floating-point arithmetic, we can parenthesize (associate) this sum in two different ways to obtain two different results :

$$\underbrace{\left( \frac{1}{\epsilon_{\text{machine}}^2} \oplus \frac{-1}{\epsilon_{\text{machine}}^2} \right)}_{=0} \oplus 1 = 1,$$
$$\frac{1}{\epsilon_{\text{machine}}^2} \oplus \underbrace{\left( \frac{-1}{\epsilon_{\text{machine}}^2} \oplus 1 \right)}_{=-1/\epsilon_{\text{machine}}^2} = 0.$$

In these notes, we will analyze the most obvious possible algorithm for  $\tilde{f}(x)$ : “naive” left-to-right summation. Later, we will see that there are often better ways to compute  $f(x)$ . A much more complete analysis of many summation algorithms can be found in Higham (1993) [1].

## 2 Naive summation

In pseudocode, the most obvious algorithm for summation of  $n$  floating-point inputs  $x \in \mathbb{F}^n$  is to simply sum the terms in order from left to right:

```
sum = 0
for i = 1 to n
    sum = sum + xi
f(x) = sum
```

For analysis, it is a bit more convenient to define the process inductively:

$$\begin{aligned} s_0 &= 0 \\ s_k &= s_{k-1} + x_k \text{ for } 0 < k \leq n, \end{aligned}$$

with  $f(x) = s_n$ . (The intermediate values  $s_k$  are known as “partial” sums.) When we implement this in floating-point arithmetic, we get the function  $\tilde{f}(x) = \tilde{s}_n$ , where  $\tilde{s}_k = \tilde{s}_{k-1} \oplus x_k$ , with  $\oplus$  denoting (correctly rounded) floating-point addition.

### 3 An upper bound on the error

We can easily prove the following upper bound on the errors accumulated by the naive floating-point sum:

$$|\tilde{f}(x) - f(x)| \leq n\epsilon_{\text{machine}} \sum_{i=1}^n |x_i| + O(\epsilon_{\text{machine}}^2). \quad (1)$$

This means that the *relative error* in the sum is bounded above by

$$\frac{|\tilde{f}(x) - f(x)|}{|f(x)|} \leq nO(\epsilon_{\text{machine}}) \left[ \frac{\sum_{i=1}^n |x_i|}{|\sum_{i=1}^n x_i|} \right]. \quad (2)$$

The  $[\dots]$  factor is what we will eventually call the **condition number** of the summation problem, a term that we will define precisely later in 18.335. In the special case of summing *nonnegative* values  $x_i \geq 0$ , the  $[\dots]$  term is  $= 1$ , and we find that the relative error grows **at worst linearly** with the problem size  $n$ .

To prove this, we first prove the lemma:

$$\tilde{f}(x) = \sum_{i=1}^n x_i \prod_{k=i}^n (1 + \epsilon_k), \quad (3)$$

where  $\epsilon_1 = 0$  and the other  $\epsilon_k$  satisfy  $|\epsilon_k| \leq \epsilon_{\text{machine}}$ , by induction on  $n$ .

- For  $n = 1$ , it is trivial with  $\epsilon_1 = 0$ .
- Now for the inductive step. Suppose  $\tilde{s}_{n-1} = \sum_{i=1}^{n-1} x_i \prod_{k=i}^{n-1} (1 + \epsilon_k)$ . Then  $\tilde{s}_n = \tilde{s}_{n-1} \oplus x_n = (\tilde{s}_{n-1} + x_n)(1 + \epsilon_n)$  where  $|\epsilon_n| < \epsilon_{\text{machine}}$  is guaranteed by floating-point addition. The result follows by inspection: the previous terms are all multiplied by  $(1 + \epsilon_n)$ , and we add a new term  $x_n(1 + \epsilon_n)$ .

Given lemma. (3), let us multiply out the terms:

$$(1 + \epsilon_1) \cdots (1 + \epsilon_n) = 1 + \sum_{k=1}^n \epsilon_k + (\text{products of } \epsilon) = 1 + \delta_i, \quad (4)$$

where the products of  $\epsilon_k$  terms are  $O(\epsilon_{\text{machine}}^2)$ , and hence

$$|\delta_i| \leq \sum_{k=i}^n |\epsilon_k| + O(\epsilon_{\text{machine}}^2) \leq n\epsilon_{\text{machine}} + O(\epsilon_{\text{machine}}^2). \quad (5)$$

Now we have:  $\tilde{f}(x) = f(x) + \sum_{i=1}^n x_i \delta_i$ , and hence (by the triangle inequality):

$$|\tilde{f}(x) - f(x)| \leq \sum_{i=1}^n |x_i| |\delta_i|.$$

Hence we obtain eq. (1) from the  $|\delta_i|$  bound above.

Note to our future selves: This does *not* correspond to a proof of forwards stability (defined soon in 18.335), since we have only shown that  $|\tilde{f}(x) - f(x)| = \|x\|_1 O(\epsilon_{\text{machine}})$ , which is different from  $|\tilde{f}(x) - f(x)| = |f(x)| O(\epsilon_{\text{machine}})$  *unless* all the  $x_i$  are  $\geq 0$ ! Our  $O(\epsilon_{\text{machine}})$  in eq. (1) is uniformly convergent in  $x$ , however (that is, the coefficient of  $\epsilon_{\text{machine}}$  is independent of  $x$ , although it depends on  $n$ ).

## 4 Average errors

In fact, the analysis above is typically too pessimistic, because the individual errors  $\epsilon_k$  are typically of *different signs*, and in particular can usually be thought of as random numbers, because the last few digits of typical inputs  $x_i$  are often random noise and IEEE arithmetic rounds-to-nearest by default. For independent random  $\epsilon_k$ , since  $\delta_i$  is the sum of  $(n-i+1)$  random variables with variance  $\sim \epsilon_{\text{machine}}$  and zero mean, it follows from the usual properties of **random walks** that the mean  $|\delta_i|$  has magnitude  $\sim \sqrt{n-i+1}O(\epsilon_{\text{machine}}) \leq \sqrt{n}O(\epsilon_{\text{machine}})$ . Hence we typically expect

$$\text{root mean square } |\tilde{f}(x) - f(x)| = O\left(\sqrt{n}\epsilon_{\text{machine}} \sum_{i=1}^n |x_i|\right),$$

i.e. rms errors that grow  $\sim \sqrt{n}$ .

This sounds good, but in fact there are summation algorithms that do **much better**. The algorithm for Julia’s built-in `sum` function, for example, is **pairwise summation**, which has  $O(\log n)$  worst-case and  $O(\sqrt{\log n})$  average-case errors [1], while having about the same performance as naive summation. There is even an algorithm called **compensated summation** that achieves  $O(1)$  error growth, albeit at increased computational cost. Nevertheless, in all of these algorithms the *relative* error is still proportional to the condition number as in eq. (2) above, so they can still give large relative errors for “badly conditioned” sums where  $|\sum x_i| \ll \sum |x_i|$ . In this case, the only way to do better is generally to increase the precision, though there are clever ways to increase the precision “adaptively”[2].

## 5 Backwards stability

We can easily adapt the error analysis above into a proof of **backwards stability** of naive summation. To be backwards stable, we must find a vector  $\tilde{x} \in \mathbb{R}^n$  such that  $\tilde{f}(x) = f(\tilde{x})$ , and also  $\tilde{x}$  is “close” to  $x$  in the sense that  $\|\tilde{x} - x\| = \|x\|O(\epsilon_{\text{machine}})$  in some norm  $\|\cdot\|$ . We do this in two steps. First, we construct  $\tilde{x}$  such that  $\tilde{f}(x) = f(\tilde{x})$ , and then we show that it is close to  $x$ .

A possible  $\tilde{x}$  follows immediately from lemma. (3) and eq. (4):

$$\tilde{x}_i = x_i \prod_{k=i}^n (1 + \epsilon_k) = x_i(1 + \delta_i).$$

Eq. (3) tells us that  $\sum_i \tilde{x}_i = f(\tilde{x})$  is equal to our naive floating-point sum  $\tilde{f}(x)$  as desired.

Then, we can use eq. (5) to show that  $\|\tilde{x} - x\|$  is small. As we shall shortly see in 18.335 (Trefethen ch. 14), it turns out that we can choose any norm that we wish for proving stability (stability in one norm implies stability in *every* norm), and in this problem it is convenient to choose the  $L_1$  norm  $\|x\|_1 = \sum_{i=1}^n |x_i|$ . In this norm, we immediately find

$$\|\tilde{x} - x\|_1 = \sum_{i=1}^n |x_i \delta_i| \leq \|x\|_1 n \epsilon_{\text{machine}} + O(\epsilon_{\text{machine}}^2) = \|x\|_1 O(\epsilon_{\text{machine}}).$$

Note that the “constant” factor  $n$  in the  $O(\epsilon_{\text{machine}})$  is independent of  $x$  (as is required for backwards stability), though it depends on  $n$ .

### 5.1 Regarding $\|x\|$ versus $\|\tilde{x}\|$

It doesn’t matter whether we use  $\|x\|$  or  $\|\tilde{x}\|$  on the right-hand/denominator for the definition of backwards stability (or stability), since by the triangle inequality  $\|x\| = \|\tilde{x} + (x - \tilde{x})\| \leq \|\tilde{x}\| + \|x - \tilde{x}\| = \|\tilde{x}\| + \|x\|O(\epsilon_{\text{machine}})$ . In particular, it follows that:

$$\|\tilde{x} - x\| = \|x\|O(\epsilon_{\text{machine}}) \iff \|\tilde{x} - x\| = \|\tilde{x}\|O(\epsilon_{\text{machine}})$$

in any norm.

## 5.2 Regarding inputs in $\mathbb{R}$ versus $\mathbb{F}$

In the beginning, we assumed that  $x$  was in  $\mathbb{F}^n$ , i.e. that the inputs are already floating point numbers. Almost the same proof applies if  $x$  is in  $\mathbb{R}^n$  and we first compute  $\text{fl}(x)$  (rounding  $x$  to the nearest floating-point values) before summing. In particular, lemma (3) is replaced with

$$\tilde{f}(x) = \sum_{i=1}^n \text{fl}(x_i) \prod_{k=i}^n (1 + \epsilon_k) = \sum_{i=1}^n x_i (1 + \epsilon) \prod_{k=i}^n (1 + \epsilon_k),$$

so that there is merely an extra  $(1 + \epsilon)$  term. This  $\epsilon$  is added to the relative error  $\delta_i$  in (4), and eq. (5) is replaced by  $|\delta_i| \leq (n+1)\epsilon_{\text{machine}} + O(\epsilon_{\text{machine}}^2)$ . Changing  $n$  to  $n+1$  in this coefficient makes no difference to our conclusions above.

## References

- [1] Nicholas J. Higham, “The accuracy of floating point summation,” *SIAM Journal on Scientific Computing* **14**, pp. 783–799 (1993).
- [2] Radford M. Neal, “Fast exact summation using small and large superaccumulators,” *arXiv:1505.05571* (2015); available in Julia via the Xsum.jl package.