1 Naive summation

In these notes, we analyze the floating-point error involved in summing \(n\) numbers, i.e. in computing the function \(f(x) = \sum_{i=1}^{n} x_i\) for \(x \in \mathbb{F}^n\) (\(\mathbb{F}\) being the set of floating-point numbers), where the sum is done in the most obvious way, in sequence. In pseudocode:

\[
\text{sum} = 0 \\
\text{for } i = 1 \text{ to } n \\
\quad \text{sum} = \text{sum} + x_i \\
\text{f}(x) = \text{sum}
\]

A much more complete analysis of summation can be found in Higham (1993) [1]. Perhaps confusingly, this naive algorithm is called “recursive” summation, in reference to the inductive version of the definition below, although most computer programs would implement this with a loop (with the exception of Lisp programmers using tail recursion).

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

\[
s_0 = 0 \\
s_k = s_{k-1} + x_k \quad \text{for } 0 < k \leq n,
\]

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.

2 An upper bound on the error

We can easily prove an upper bound on the errors accumulated by the floating-point implementation of this algorithm:

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

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

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

The \([\ldots]\) 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 \([\ldots]\) term is \(= 1\), and we find that the relative error grows at worse linearly with the problem size \(n\).

To prove this, we first prove the lemma:

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

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 $x_n(1 + \epsilon_n)$.

Now, let us multiply out the terms:

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

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

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

Now we have: $\hat{f}(x) = f(x) + (x_1 + x_2)\delta_2 + \sum_{i=3}^{n} x_i \delta_i$, and hence (by the triangle inequality):

$$|\hat{f}(x) - f(x)| \leq |x_1| |\delta_2| + \sum_{i=2}^{n} |x_i| |\delta_i|.$$

Hence $|\hat{f}(x) - f(x)| \leq n\epsilon_{\text{machine}} \sum_{i=1}^{n} |x_i|$ from the $|\delta_i|$ bound above.

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

3 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 usually random noise. For uniform random $\epsilon_k$, since $\delta_i$ is the sum of $(n - i + 1)$ random variables with variance $\sim \epsilon_{\text{machine}}^2$, 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

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

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 $\text{sum}$ 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.

References
